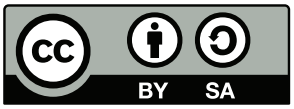
Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Uma infraestrutura baseada em certificados digitais para efetuar autenticação de cliente

Leonardo Schäffer

Supervisor:
Prof. Dr. Alfredo Goldman vel Lejbman

São Paulo
Dezembro de 2016



Agradecimentos

Gostaria de agradecer ao meu supervisor, Alfredo, e ao meu colega de Nubank, Jonas Abreu, sem os quais este trabalho não teria sido possível.

Gostaria, também, de dedicar este trabalho e agradecer aos meus pais, Gerd e Silvana, por tudo.

Resumo

Segurança da informação é fundamental para qualquer organização que oferece serviços pela internet que exigem confidencialidade e integridade dos dados. Este trabalho apresenta uma solução para o acesso seguro a serviços online baseada em autenticação de clientes utilizando de certificados digitais e TLS mútuo, com o objetivo de prevenção a fraudes financeiras. São introduzidos conceitos de segurança, criptografia (em especial criptografia de chave pública), infraestrutura de chave pública e o protocolo HTTPS (HTTP sobre TLS), dos quais a solução depende. Por fim, são discutidas as vantagens e desvantagens da solução e é descrito o processo de desenvolvimento e implementação dela no Nubank, uma emissora de cartões de crédito, cuja infraestrutura se baseia em microserviços rodando em Amazon Web Services, e cujos clientes utilizam tais serviços através de aplicativos móveis.

Palavras-chave: segurança, certificado, autoridade certificadora, CA, infraestrutura de chave pública, TLS, SSL, HTTPS, autenticação de cliente.

Abstract

Information security is fundamental for every organization that offers internet services that need confidentiality and data integrity. This work introduces a solution for the secure access of online services based on client authentication, through digital certificates and mutual TLS, having financial frauds prevention as its main objective. Concepts of security, criptography (mostly public key criptography), public key infrastructure and the protocol HTTPS (HTTP over TLS) of which the solution depends, are introduced. Finally, advantages and disadvantages of the solution are discussed and development and implementation process at Nubank is described. Nubank is a credit card issuer, its infrastructure is based on microservices running on Amazon Web Services, and its clients use the services through their mobile applications.

Keywords: security, certificate, certificate authority, CA, public key infrastructure, TLS, SSL, HTTPS, client authentication.

Sumário

Capítulo 1

Introdução

Em muitas organizações, em especial instituições financeiras, fraudes fazem parte do dia-a-dia, e podem causar grandes prejuízos. Esse é um problema do qual não é viável se proteger por completo, inevitavelmente uma ou outra tentativa de fraude é bem sucedida.

Para combater fraudes, é necessário se defender por duas estratégias diferentes, falando a grosso modo. Primeiro, minimizar o prejuízo que uma única fraude pode causar. Uma das maneiras mais básicas de se combater isso é impondo limites a operações. Por exemplo, limite de crédito. Segundo, cortar classes inteiras de fraudes através de mecanismos e processos bem desenvolvidos.

Processos de combate a muitas fraudes podem ser feitos sem o uso de tecnologia, tendo um humano por trás, revisando e controlando certas operações, não as aprovando quando houver suspeita de fraude. Porém, basear o combate a fraudes em fiscalização humana é muito lento e burocrático para o cliente, além do fato de ser extremamente custoso, pois seriam necessários muitos funcionários. Com o uso de tecnologia, não é possível cortar todo o processo de fiscalização humana, mas é possível diminuir enormemente esse trabalho.

O Nubank é uma instituição financeira emissora de cartões de crédito. Toda a interação do cliente com os serviços da empresa acontece através de aplicativos para as 3 principais plataformas móveis, Android, iPhone e Windows Phone.

Nesse trabalho o objetivo é aumentar o nível de segurança em geral para clientes do Nubank e minimizar prejuízos causados por fraudes, usando a segunda estratégia, de cortar classes inteiras de fraude, e através de uma maneira eficiente com o uso de tecnologia.

Mais especificamente, o objetivo é garantir que operações perigosas por parte do cliente (como movimentar dinheiro, por exemplo), venham apenas dos aplicativos, dificultando diversos tipos de ataques comuns em sistemas não móveis e navegadores web, como XSS (*Cross-site Scripting*) [22], CSRF

(*Cross-Site Request Forgery*) [23], e vários tipos de *spoofing*. Dessa forma, qualquer fraude que dependa do acesso às contas dos clientes se torna muito difícil de ser realizado, talvez até difícil o suficiente para que não valha a pena para o fraudador, e ele desista por completo.

Outro objetivo é também ter um mecanismo auditável, que possa ser legalmente, de se provar que o cliente realizou uma determinada operação.

A solução encontrada foi realizar autenticação de cliente, utilizando de certificados digitais e TLS (*Transport Layer Security*) mútuo. Trata-se de um esquema parecido com autenticação de dois fatores (*two-factor authentication*), porém com outras implicações práticas.

Essa solução diminui bastante a chance da conta do cliente, no aplicativo para celular, ter um acesso indevido, dando uma confiança maior para o Nubank de que as operações realizadas foram de fato feitas pelos donos das contas.

Ela também pode ser utilizada na infraestrutura interna, para proteger o acesso dos funcionários aos serviços, com o mesmo fim, evitando também muitas fraudes baseadas no comprometimento de mecanismos internos da empresa.

1.1 Organização do texto

Esse trabalho está dividido em duas partes, uma teórica (capítulos 2 a 5), que apresenta conceitos de segurança e criptografia. E uma parte prática (capítulo 6), que conta sobre a solução desenvolvida para realização de autenticação de cliente.

- No capítulo 2 são introduzidos alguns conceitos básicos de segurança necessários para o entendimento do resto do texto.
- No capítulo 3 é explicado um pouco de criptografia, em especial criptografia de chave pública.
- O capítulo 4 explica o que é infraestrutura de chave pública, que funciona a partir de certificados digitais e autoridades certificadoras.
- O capítulo 5 fala sobre TLS, o protocolo criptográfico do qual esse trabalho depende.
- O capítulo 6 explica em detalhes a solução desenvolvida e o seu processo de implementação.

Capítulo 2

Segurança da informação

Segurança da informação é uma ampla área de estudo, que além de computação (implementação), envolve matemática (criptografia), engenharia (dispositivos), economia (impacto financeiro) e psicologia (como pessoas interagem com mecanismos de segurança).

2.1 Criptografia e Segurança

Criptografia e segurança não são a mesma coisa. Segurança, falando de maneira geral, utiliza de criptografia, depende de criptografia, e portanto está um nível acima, pensando em camadas de abstração.

2.2 É possível provar que um sistema é completamente seguro?

Um dos principais motivos de segurança ser um grande desafio, é o fato de que é impossível provar que um mecanismo de segurança é completamente seguro [21]. Nada garante que um sistema considerado seguro hoje não possua falhas de segurança ainda não descobertas.

Na prática, um algoritmo/protocolo é considerado suficientemente seguro, depois que vários especialistas gastaram bastante tempo tentando diversas maneiras diferentes de encontrar uma brecha nele, uma maneira de o burlar que não envolva força bruta.

2.3 Segurança versus Usabilidade

Segurança versus usabilidade é o principal *trade-off* existente em segurança da informação. É normal um aumento da segurança de um sistema atrapalhar a usabilidade.

Todo sistema que possui usuários, por mais seguro que seja, precisa de uma porta de entrada para esse usuário. Toda informação que está protegida, encriptada, precisa de uma maneira de ser decriptada para ser visualizada, por exemplo.

Esses mesmos mecanismos de acesso podem ser explorados por atacantes, inclusive esse é o método de ataque mais eficiente atualmente, roubo de credenciais [17]. É mais rápido e barato roubar credenciais, utilizando, por exemplo, de engenharia social, do que estudar um sistema para tentar encontrar uma brecha.

Mecanismos que tentam proteger esse vetor de ataque diminuem a usabilidade do sistema. Por exemplo, controle de acesso é ótimo para dar permissão a algum recurso do sistema a um usuário específico apenas quando ele precisa daquele acesso. Porém, esse usuário vai precisar obter esse acesso de alguma maneira provavelmente não automática, o que costuma ser demorado, atrapalhando a usabilidade.

Outro bom exemplo é autenticação de dois fatores, ela aumenta bastante a segurança para uma conta de usuário em um sistema, pois para o atacante não basta saber a senha do usuário para ganhar acesso. Mas também diminui a usabilidade, pois o usuário, ao fazer *login*, além de digitar a senha, precisa pegar o seu celular para ver o código que deve ser digitado para completar a operação, o que é um inconveniente.

Por outro lado, existe muito espaço para inovação na área de *design* de mecanismos de segurança, maneiras estão sendo buscadas para conseguir vencer em parte esse *trade-off*. É uma mudança de paradigma que está acontecendo entre profissionais da área [19] [20].

2.4 Segurança através de obscuridade

Um sistema faz segurança através de obscuridade quando utiliza de algoritmos de criptografia ou protocolos de segurança que não são públicos. Em um primeiro momento, isso até parece ser uma boa ideia. Manter os algoritmos/protocolos secretos parece tornar mais difícil de quebrá-los ou burlá-los.

Porém, na prática, não é isso que acontece. É extremamente difícil desenvolver tanto algoritmos de criptografia quanto protocolos de segurança que não possuam falhas, pelo fato de que existem diversos tipos de ataques diferentes. E os algoritmos/protocolos desenvolvidos precisam ser fortes contra todos.

Isso é contraintuitivo, mas os melhores algoritmos e protocolos são abertos e bem conhecidos. Eles são seguros, pois diversos especialistas participaram do seu *design* e tentaram diversos ataques para quebrá-los, sem sucesso.

O mesmo vale para implementações de algoritmos/protocolos; não é qualquer um que deve implementá-los, pois existem diversos ataques em cima de

implementações. O OpenSSL por exemplo já sofreu com várias falhas por conta de manipulação errada de *buffers*, foi o caso do infame Heartbleed [24]. Também existem ataques mais genéricos, como por exemplo ataques de tempo [25].

Utilizar algoritmos e protocolos abertos e bastante estudados ao invés de se fazer segurança através de obscuridade é um forte consenso entre especialistas da área [18].

2.5 Conceitos de segurança

A seguir são introduzidos alguns conceitos de segurança necessários para o entendimento do resto do texto.

Segredo é toda e qualquer informação que se deseja manter confidencial.

Confidencialidade é a propriedade de proteger o acesso a determinados dados por agentes não autorizados.

Encriptação é uma maneira de se garantir confidencialidade. É o ato de transformar, utilizando de uma chave e um algoritmo de encriptação, o conteúdo de uma mensagem em dados que não consigam ser lidos por agentes não autorizados, apenas quem possuir a chave consegue.

Integridade é a propriedade de garantir que o conteúdo de uma mensagem não possa ser modificado por terceiros antes de chegar em seu destino sem que isso seja identificado. Costuma ser essencial na transmissão de mensagens encriptadas, pois sem uma forma de garantir integridade, mesmo que um intermediário não consiga lê-la, ele pode corrompê-la.

Autenticação é quando um agente consegue verificar a identidade de outro agente. É importante fazer autenticação da outra parte antes de iniciar uma comunicação, para garantir que se está comunicando com a pessoa certa.

Autenticação sozinha não necessariamente garante o acesso a recursos de um sistema. **Autorização** é um passo extra que verifica e concede permissão de uso a um recurso para um usuário já autenticado.

A propriedade de **não-repúdio** é um passo extra em relação a autenticidade, confidencialidade e integridade. Não-repúdio garante que um usuário único efetuou uma determinada operação, e que esse usuário não consegue negar que a fez. Garanti-lo é essencial para operações financeiras, inclusive por conta de questões legais.

Capítulo 3

Criptografia

O objetivo desse capítulo é, sem entrar em detalhes matemáticos, dar uma rápida introdução a criptografia, em especial a criptografia de chave pública. Pois certificados digitais e autoridades certificadoras, que são a base para a solução apresentada nesse trabalho, funcionam em cima disso.

Ao se utilizar de criptografia, é mais importante entender o que ela faz e não faz, do que entender como ela funciona. E saber como utilizá-la da maneira correta. Infelizmente é muito fácil utilizar algoritmos de criptografia matematicamente corretos da maneira errada, inclusive, bibliotecas de criptografia costumam ter péssimas APIs, como por exemplo o OpenSSL, uma das mais utilizadas.

3.1 Criptografia simétrica

Criptografia simétrica ou criptografia de chave única se refere a método de encriptação em que só há uma chave envolvida, que encripta e decripta as mensagens. Portanto duas partes que queiram se comunicar com criptografia simétrica para ter confidencialidade necessitam ambas possuir a mesma chave.

AES (*Advanced Encryption Standard*), que foi publicado pelo NIST (*National Institute of Standards and Technology*) em 2001, é um dos algoritmos mais utilizados, ele aceita chaves de 128, 192 ou 256 *bits* de comprimento.

Algoritmos de chave simétrica costumam ser bem mais rápidos do que encriptação de chave pública, por isso o TLS os utiliza para encriptar as mensagens trocadas.

3.2 Função *hash* criptográfica

Funções *hash* criptográficas são funções aplicadas em cima de uma mensagem que geram um valor *hash*, uma sequência de *bytes* que identifica a mensagem, e portanto pode ser usada para verificar sua integridade.

Suas principais propriedades são que é computacionalmente inviável descobrir qual é a mensagem a partir de seu *hash* e que é inviável descobrir duas mensagens que possuem o mesmo *hash* (ou seja, a função é segura contra colisões).

A família de funções hash criptográficas SHA (*Secure Hash Algorithm*) [15] possui as funções mais utilizadas. É publicada pelo NIST.

Funções de *hash* são muito mais rápidas de calcular do que assinaturas digitais, apresentadas na próxima seção.

3.3 *Message Authentication Code*

MACs oferecem integridade e autenticação de mensagens.

Podem ser construídos a partir de funções de *hash* criptográficas, nesse caso são chamados de HMAC (*hash-based message authentication code*).

O TLS utiliza MACs para garantir a integridade das mensagens trocadas.

3.4 Criptografia de chave pública

Diferentemente de criptografia simétrica, em criptografia de chave pública, cada parte possui duas chaves, uma pública e uma privada.

A chave pública, como o nome diz, pode ser acessível publicamente, ela é usada para encriptar mensagens.

Já a chave privada deve ser acessível apenas pelo dono do par de chaves, ela é usada para decriptar mensagens encriptadas com sua chave pública.

Essa é a grande vantagem de criptografia de chave pública, a chave privada não precisa ser compartilhada, e portanto não precisa transitar por nenhum meio de comunicação. Ela pode permanecer o tempo todo no dispositivo onde foi gerada.

Se não fosse criptografia de chave pública, fazer comunicação com segurança na internet seria extremamente mais complicado.

3.4.1 Assinaturas digitais

Outro uso para pares de chaves é a realização de assinaturas digitais. Uma assinatura digital se trata de uma sequência de *bytes* gerada através de uma operação matemática que tem a chave privada e a mensagem a ser assinada como parâmetros de entrada.

O destinatário da mensagem acompanhada de sua assinatura pode utilizar a chave pública do remetente para verificar a validade da assinatura.

Dessa forma assinaturas digitais são utilizadas para se garantir integridade e não-repúdio, e fazer autenticação de mensagens. Por conta dessas propriedades, assinaturas digitais podem ser utilizadas por processos que tenham valor legal.

3.4.2 Troca de chaves Diffie-Hellman

A troca de chave Diffie-Hellman foi um dos primeiros protocolos de chave pública a ser criado. É usado por duas partes para a obtenção de um segredo compartilhado através da troca de mensagens por um meio de comunicação inseguro.

Normalmente, o propósito do segredo é ser usado como chave para realizar encriptação de mensagens usando um algoritmo de chave simétrica.

Diffie-Hellman é utilizado pelo TLS para trocar a chave de cada sessão.

3.4.3 RSA

RSA é o primeiro criptossistema prático de chave pública a ser inventado, foi publicado em 1977. Sua segurança se baseia no problema de calcular a fatoração do produto de dois grandes números primos. O nome RSA é formado a partir das inciais dos sobrenomes de seus criadores, Ron Rivest, Adi Shamir, e Leonard Adleman.

Chaves RSA devem ter entre 2048 e 4096 *bits* de comprimento.

Pelo fato de até hoje ainda não ter sido quebrado, continua sendo vastamente utilizado.

3.4.4 Criptografia de curvas elípticas

Posterior ao RSA, foram desenvolvidas técnicas de criptografia baseadas em curvas elípticas. Sua segurança está baseada em outro problema matemático.

Seu principal atrativo, é que as chaves podem ser muito menores do que chaves RSA, garantindo o mesmo nível de segurança. Dessa maneira diminuindo bastante a quantidade de *bytes* de chaves que precisam ser armazenados e transferidos.

Sua única grande desvantagem perante RSA é que a verificação de assinaturas é mais lenta.

Algoritmos de curvas elípticas estão sendo, lentamente, cada vez mais adotados.

3.5 Algoritmos seguros

É importante escolher com cuidado os algoritmos criptográficos que serão usados. Alguns algoritmos tem falhas críticas descobertas e se tornam obsoletos.

O NIST possui boas recomendações sobre quais algoritmos utilizar em [14].

3.6 Comprimento de chave

Outro fator importante ao se utilizar criptografia é escolher o comprimento das chaves utilizadas, ou seja, quantos *bits* terão.

O principal fator que governa a escolha é por quanto tempo a chave deverá ser válida, quanto mais tempo, maior deve ser. É importante prestar atenção nisso ao gerenciar chaves.

O cálculo é baseado no tempo que leva para a chave ser quebrada por força bruta. Lembrando que os valores mudam a medida que CPUs e GPUs ficam mais rápidas e técnicas de quebra são aperfeiçoadas.

O NIST possui boas recomendações sobre quais comprimentos de chave usar em [14].

3.7 *Secret sharing*

Secret sharing é uma técnica criptográfica que permite que um segredo seja quebrado em n partes, de maneira que $m \leq n$ partes são necessárias para recompô-lo.

Uma prática comum é dividir as partes entre pessoas de confiança, outra é guardar cada parte em um local diferente, assim diminuindo consideravelmente a possibilidade do segredo ser comprometido.

Uma implementação popular é a SSSS (*Shamir's Secret Sharing Scheme*) [16], baseado na técnica de interpolação polinomial, inventado por Adi Shamir.

SSSS tem como limitação o tamanho do segredo, aceita apenas até 128 *bytes*. Nesse caso, é possível gerar uma chave aleatória até esse tamanho, para utilizar com uma cifra simétrica para encriptar o segredo. E então em seguida aplicar *secret sharing* sobre a chave aleatória.

SSSS é utilizado para proteger a chave privada do certificado raiz do Nubank.

Capítulo 4

Infraestrutura de chave pública

Uma infraestrutura de chave pública (PKI) administra certificados digitais e criptografia de chave pública, com o objetivo de viabilizar comunicação eletrônica segura.

Seus componentes funcionais são:

- Uma autoridade certificadora (CA, *Certificate Authority*), que emite e revoga certificados.
- Uma autoridade registradora (RA, *Registration Authority*), que verifica a identidade de entidades que requisitam certificados.
- Um diretório central para armazenar e indexar certificados e listas de revogação.

Organizações que executam a função de CA costumam executar as outras funções também.

Abaixo, certificados digitais e CAs são explicadas em detalhe.

4.1 Certificados digitais

Certificados digitais são necessários para se realizar autenticação de cliente com TLS, assunto do próximo capítulo.

De maneira resumida, um certificado digital contém um nome (que identifica a quem pertence), uma chave pública de seu dono e uma assinatura efetuada por uma CA.

O propósito do certificado é certificar que a chave pública do certificado realmente pertence ao dono do certificado, e que portanto, pode ser utilizada para se comunicar com ele.

Certificados seguem o padrão X.509 (atualmente na versão 3), especificado no RFC 5280 [1], e contém muitos outros campos com informações adicionais, que estão descritos abaixo.

4.1.1 Formato

O formato X.509 é definido através de ASN.1 (*Abstract Syntax Notation One*) [12], um padrão de notação formal para descrever a sintaxe de tipos de dados e valores.

Certificados possuem os seguintes campos:

- ***Version:*** Indica a versão do padrão X.509 usado, normalmente é a versão 3.
- ***Serial number:*** O número de série que identifica o certificado, é este número que é utilizado na composição de listas de revogação.
- ***Signature algorithm:*** O algoritmo utilizado para assinar o certificado, o mais comum atualmente é o SHA256.
- ***Issuer:*** O emissor desse certificado, ou seja, quem o assina. Mais especificamente, o campo "*Subject*" deste.
- ***Validity:*** Informa o período para o qual o certificado é válido.
- ***Subject:*** Um nome que identifica o dono desse certificado. Esse nome segue o padrão X.500 (utilizado pelo LDAP), especificado no RFC 1779 [2].

  No exemplo da próxima seção tem-se "Subject: C=US, ST=California, L=Los Angeles, O=Internet Corporation for Assigned Names and Numbers, OU=Technology, CN=www.example.org", que significam o seguinte:

  - **C:** *Country* (país).
  - **ST:** *State* (estado).
  - **L:** *Localization* (localização).
  - **O:** *Organization* (organização).
  - **OU:** *Organization unit* (unidade da organização).
  - **CN:** *Common name* (nome comum).
- ***Subject Public Key Info:*** A chave pública desse certificado e um identificador de seu algoritmo.
- ***Signature Algorithm:*** Um identificador do algoritmo usado para assinar o certificado, e sua assinatura, calculada sobre todas as outras informações do certificado.

- **X509v3** ***extensions:*** Extensões do certificado. O Padrão X.509 permite que qualquer um crie extensões para serem embutidas nos certificados, isso é bastante interessante, pois permite o desenvolvimento de soluções ad-hoc.

  Algumas extensões estão definidas no próprio RFC do X.509 [1].

  Abaixo estão descritas algumas das extensões mais importantes:

  - ***Basic Constraints:*** Este campo indica se o certificado é um certificado de uma CA, ou seja, se ele pode assinar outros certificados ou não. É um campo obrigatório.
  - ***Key Usage e Extended Key Usage:*** Indica quais os usos permitidos para a chave do certificado.

    Principais usos:

    * ***Digital Signature:*** Realizar assinaturas digitais.
    * ***Key Encipherment:*** Encriptar outras chaves.
    * ***Data Encipherment:*** Encriptar dados.
    * ***Key Agreement:*** Realizar troca de chaves efêmeras.
    * ***Certificate Sign:*** Assinar outros certificados.
    * ***CRL Sign:*** Assinar listas de revogação.
    * ***TLS Web Server Authentication:*** Autenticação de servidor HTTPS.
    * ***TLS Web Client Authentication:*** Autenticação de cliente HTTPS.
  - ***CRL Distribution Points:*** Endereços de onde obter listas de revogação.
  - ***Subject Alternative Names:*** Outros nomes para os quais o certificado é válido.

    Esse campo é bastante importante para certificados de servidores HTTPS. Ele deve conter todos os nomes de domínio para o qual o certificado é válido. Clientes ao conectar em um domínio que apresenta este certificado, devem checar se este domínio está contido no certificado, e se não estiver, devem invalidar a conexão.

4.1.2 Exemplo

Segue abaixo o certificado do domínio example.org exibido através do comando openssl x509 -text -noout -in certificado.pem.

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
```

```
    0e:64:c5:fb:c2:36:ad:e1:4b:17:2a:eb:41:c7:8c:b0
Signature Algorithm: sha256WithRSAEncryption
  Issuer: C=US, O=DigiCert Inc, OU=www.digicert.com,
    CN=DigiCert SHA2 High Assurance Server CA
  Validity
    Not Before: Nov  3 00:00:00 2015 GMT
    Not After : Nov 28 12:00:00 2018 GMT
  Subject: C=US, ST=California, L=Los Angeles,
    O=Internet Corporation for Assigned Names and Numbers,
    OU=Technology, CN=www.example.org
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
      Modulus:
        00:b3:40:96:2f:61:63:3e:25:c1:97:ad:65:45:fb:
        ef:13:42:b3:2c:99:86:f4:b5:80:0b:76:dc:06:38:
        2c:1f:a3:62:55:5a:36:76:de:ae:5d:fc:e2:e5:b4:
        e6:ec:5d:ca:ee:ca:df:50:16:24:2c:ee:fc:9a:b6:
        8c:f6:a8:b3:ac:7a:08:7b:2a:1f:ad:5f:e7:fa:96:
        59:25:ab:90:b0:f8:c2:3f:13:04:26:74:68:0f:c6:
        78:2a:95:8a:5f:42:f2:0e:ed:52:a6:eb:68:23:89:
        e5:43:f8:6d:12:1b:62:42:7b:a8:05:f3:59:c4:5e:
        d6:c5:cc:46:c0:4b:19:b9:2d:4a:71:72:24:1e:5e:
        55:44:93:ab:78:a1:47:4d:a5:dc:07:5a:9c:67:f4:
        11:68:12:2f:d3:28:71:bc:ad:72:05:3c:16:75:d4:
        f8:72:58:ba:19:f1:dc:09:ed:f1:18:c6:92:2f:7d:
        bc:16:0b:37:8d:8a:ef:1b:6f:4f:b9:e0:7a:54:98:
        bf:b5:b6:cf:bb:aa:93:7f:0a:7f:1f:56:eb:a9:d8:
        e1:db:d5:39:d8:18:5b:d1:f2:64:33:d0:d6:c4:23:
        ff:09:ab:6d:71:ce:da:cf:c1:17:9c:23:be:2c:af:
        2f:92:1c:3f:90:08:89:58:f2:b1:e1:10:6f:83:2e:
        f7:9f
      Exponent: 65537 (0x10001)
  X509v3 extensions:
    X509v3 Authority Key Identifier:
      keyid:51:68:FF:90:AF:02:07:75:3C:CC:D9:65:64:62:A2:12:B8:59:72:3B

    X509v3 Subject Key Identifier:
      A6:4F:60:1E:1F:2D:D1:E7:F1:23:A0:2A:95:16:E4:E8:9A:EA:6E:48
    X509v3 Subject Alternative Name:
      DNS:www.example.org, DNS:example.com, DNS:example.edu, DNS:example.net,
      DNS:example.org, DNS:www.example.com, DNS:www.example.edu,
      DNS:www.example.net
    X509v3 Key Usage: critical
      Digital Signature, Key Encipherment
    X509v3 Extended Key Usage:
      TLS Web Server Authentication, TLS Web Client Authentication
    X509v3 CRL Distribution Points:

      Full Name:
        URI:http://crl3.digicert.com/sha2-ha-server-g4.crl

      Full Name:
        URI:http://crl4.digicert.com/sha2-ha-server-g4.crl

    X509v3 Certificate Policies:
      Policy: 2.16.840.1.114412.1.1
        CPS: https://www.digicert.com/CPS
      Policy: 2.23.140.1.2.2

    Authority Information Access:
      OCSP - URI:http://ocsp.digicert.com
```

```
        CA Issuers -
          URI:http://cacerts.digicert.com/DigiCertSHA2HighAssuranceServerCA.crt

      X509v3 Basic Constraints: critical
        CA:FALSE
  Signature Algorithm: sha256WithRSAEncryption
     84:a8:9a:11:a7:d8:bd:0b:26:7e:52:24:7b:b2:55:9d:ea:30:
     89:51:08:87:6f:a9:ed:10:ea:5b:3e:0b:c7:2d:47:04:4e:dd:
     45:37:c7:ca:bc:38:7f:b6:6a:1c:65:42:6a:73:74:2e:5a:97:
     85:d0:cc:92:e2:2e:38:89:d9:0d:69:fa:1b:9b:f0:c1:62:32:
     65:4f:3d:98:db:da:d6:66:da:2a:56:56:e3:11:33:ec:e0:a5:
     15:4c:ea:75:49:f4:5d:ef:15:f5:12:1c:e6:f8:fc:9b:04:21:
     4b:cf:63:e7:7c:fc:aa:dc:fa:43:d0:c0:bb:f2:89:ea:91:6d:
     cb:85:8e:6a:9f:c8:f9:94:bf:55:3d:42:82:38:4d:08:a4:a7:
     0e:d3:65:4d:33:61:90:0d:3f:80:bf:82:3e:11:cb:8f:3f:ce:
     79:94:69:1b:f2:da:4b:c8:97:b8:11:43:6d:6a:25:32:b9:b2:
     ea:22:62:86:0d:a3:72:7d:4f:ea:57:3c:65:3b:2f:27:73:fc:
     7c:16:fb:0d:03:a4:0a:ed:01:ab:a4:23:c6:8d:5f:8a:21:15:
     42:92:c0:34:a2:20:85:88:58:98:89:19:b1:1e:20:ed:13:20:
     5c:04:55:64:ce:9d:b3:65:fd:f6:8f:5e:99:39:21:15:e2:71:
     aa:6a:88:82
```

4.1.3 Formato de arquivo

Existem dois principais formatos para certificados, DER e PEM.

DER (*Distinguished Encoding Rules*), é serialização binária do formato ASN.1.

PEM, é a conversão do DER para Base64, tornando-o um formato fácil para ser transmitido como texto puro, por exemplo, em respostas a requisições HTTP. Além do Base64, o PEM contém um cabeçalho e um rodapé que indicam o início e o fim do bloco relativo ao certificado.

Exemplo

O arquivo PEM do mesmo certificado do domínio example.org que foi exibido acima:

```
-----BEGIN CERTIFICATE-----
MIIF8jCCBNqgAwIBAgIQDmTF+8I2reFLFyrrQceMsDANBgkqhkiG9w0BAQsFADBw
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMS8wLQYDVQQDEyZEaWdpQ2VydCBTSEEyIEhpZ2ggQXNz
dXJhbmNlIFNlcnZlciBDQTAeFw0xNTExMDMwMDAwMDBaFw0xODExMjgxMjAwMDBa
MIGlMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxML
TG9zIEFuZ2VsZXMxPDA6BgNVBAoTM0ludGVybmV0IENvcnBvcmF0aW9uIGZvciBB
c3NpZ25lZCBOYW1lcyBhbmQgTnVtYmVyczETMBEGA1UECxMKVGVjaG5vbG9neTEY
MBYGA1UEAxMPd3d3LmV4YW1wbGUub3JnMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A
MIIBCgKCAQEAs0CWL2FjPiXBl61lRfvvE0KzLJmG9LWAC3bcBjgsH6NiVVo2dt6u
Xfzi5bTm7F3K7srfUBYkLO78mraM9qizrHoIeyofrV/n+pZZJauQsPjCPxMEJnRo
D8Z4KpWKX0LyDu1SputoI4nlQ/htEhtiQnuoBfNZxF7WxcxGwEsZuS1KcXIkHl5V
RJOreKFHTaXcB1qcZ/QRaBIv0yhxvK1yBTwWddT4cli6GfHcCe3xGMaSL328Fgs3
jYrvG29PueB6VJi/tbbPu6qTfwp/H1brqdjh29U52Bhb0fJkM9DWxCP/Cattcc7a
z8EXnCO+LK8vkhw/kAiJWPKx4RBvgy73nwIDAQABo4ICUDCCAkwwHwYDVR0jBBgw
FoAUUWj/kK8CB3U8zNllZGKiErhZcjswHQYDVR0OBBYEFKZPYB4fLdHn8SOgKpUW
5Oia6m5IMIGBBgNVHREEejB4gg93d3cuZXhhbXBsZS5vcmeCC2V4YW1wbGUuY29t
ggtleGFtcGxlLmVkdYILZXhhbXBsZS5uZXSCC2V4YW1wbGUub3Jngg93d3cuZXhh
bXBsZS5jb22CD3d3dy5leGFtcGxlLmVkdYIPd3d3LmV4YW1wbGUubmV0MA4GA1Ud
DwEB/wQEAwIFoDAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwdQYDVR0f
```

```
BG4wbDA0oDKgMIYuaHR0cDovL2NybDMuZGlnaWNlcnQuY29tL3NoYTItaGEtc2Vy
dmVyLWc0LmNybDA0oDKgMIYuaHR0cDovL2NybDQuZGlnaWNlcnQuY29tL3NoYTIt
aGEtc2VydmVyLWc0LmNybDBMBgNVHSAERTBDMDcGCWCGSAGG/WwBATAqMCgGCCsG
AQUFBwIBFhxodHRwczovL3d3dy5kaWdpY2VydC5jb20vQ1BTMAgGBmeBDAECAjCB
gwYIKwYBBQUHAQEEdzB1MCQGCCsGAQUFBzABhhhodHRwOi8vb2NzcC5kaWdpY2Vy
dC5jb20wTQYIKwYBBQUHMAKGQWh0dHA6Ly9jYWNlcnRzLmRpZ2ljZXJ0LmNvbS9E
aWdpQ2VydFNIQTJIaWdoQXNzdXJhbmNlU2VydmVyQ0EuY3J0MAwGA1UdEwEB/wQC
MAAwDQYJKoZIhvcNAQELBQADggEBAISomhGn2L0LJn5SJHuyVZ3qMIlRCIdvqe0Q
6ls+C8ctRwRO3UU3x8q8OH+2ahxlQmpzdC5al4XQzJLiLjiJ2Q1p+hub8MFiMmVP
PZjb2tZm2ipWVuMRM+zgpRVM6nVJ9F3vFfUSHOb4/JsEIUvPY+d8/Krc+kPQwLvy
ieqRbcuFjmqfyPmUv1U9QoI4TQikpw7TZU0zYZANP4C/gj4Ry48/znmUaRvy2kvI
l7gRQ21qJTK5suoiYoYNo3J9T+pXPGU7Lydz/HwW+w0DpArtAaukI8aNX4ohFUKS
wDSiIIWIWJiJGbEeIO0TIFwEVWTOnbNl/faPXpk5IRXicapqiII=
-----END CERTIFICATE-----
```

PEM também é utilizado para outras informações, como chaves públicas e privadas. Além disso, um mesmo arquivo PEM pode conter múltiplos certificados e chaves, por isso a necessidade do cabeçalho e rodapé.

4.2 Revogação de certificados

Certificados possuem uma data de validade. Porém, é possível que um certificado seja comprometido antes de expirar, quando, por exemplo, a chave privada é exposta ou roubada. Quando isso acontece é necessário que o certificado não seja mais aceito por terceiros. Isso é chamado de revogação de certificado.

Infelizmente não existe nenhuma mágica de criptografia para se cuidar disso. Revogação acontece através da manutenção de listas de certificados revogados.

Esse é um dos pontos negativos em infraestrutura de chave pública. A dificuldade em se manter as listas.

Existem duas maneiras principais de se realizar revogação: CRLs e OCSP.

4.2.1 *Certificate Revocation Lists*

CRL (*Certificate Revocation List*) trata-se de, literalmente, uma lista. No processo de verificação de um certificado, é necessário obter as listas de revogação e verificar se o número de série do certificado está contido nelas ou não.

A CRL segue um formato padronizado, descrito com ASN.1, e está especificada no mesmo RFC do X.509 [1].

Entre as informações importantes que a CRL contém estão:

- ***Issuer:*** O emissor dessa CRL.
- ***Next Update:*** A data na qual é prometido haver uma nova atualização da lista.

- ***Revoked Certificates:*** A lista de certificados revogados, contendo o número de série (*Serial Number*) e a data de revogação (*Revocation Date*).

- ***Signature Algorithm:*** Um identificador do algoritmo usado para assinar a CRL, e sua assinatura.

Exemplo

Segue um exemplo de CRL exibida através do comando openssl crl -text -noout -in crl.pem. Como a lista é muito grande, foram omitidos a maioria dos certificados onde há "(...)":

```
Certificate Revocation List (CRL):
    Version 2 (0x1)
  Signature Algorithm: sha256WithRSAEncryption
    Issuer: /C=US/O=DigiCert Inc/OU=www.digicert.com/
      CN=DigiCert SHA2 High Assurance Server CA
    Last Update: Oct 21 17:13:22 2016 GMT
    Next Update: Oct 28 17:00:00 2016 GMT
    CRL extensions:
      X509v3 Authority Key Identifier:
        keyid:51:68:FF:90:AF:02:07:75:3C:CC:D9:65:64:62:A2:12:B8:59:72:3B

      X509v3 CRL Number:
        554
      X509v3 Issuing Distrubution Point: critical
        Full Name:
          URI:http://crl3.digicert.com/sha2-ha-server-g4.crl

Revoked Certificates:
  Serial Number: 017B944CBD5E817FBE66158B5B0E0747
    Revocation Date: Apr 21 19:28:18 2015 GMT
  Serial Number: 09C573E6C7756DF77C78FC6E51BEB7DD
    Revocation Date: Apr 21 20:27:02 2015 GMT
  (...)
  Serial Number: 06C76F24F17D0B9D59B8289AC34C554B
    Revocation Date: Oct 21 16:16:08 2016 GMT
  Signature Algorithm: sha256WithRSAEncryption
     2e:60:d8:3f:49:f5:a9:a1:39:d2:1d:68:b4:ec:35:31:b1:68:
     6a:36:e7:c7:d2:96:2c:35:93:76:19:4e:c6:e0:ac:5a:d9:0a:
     77:7b:4a:af:dd:7b:06:ca:af:44:2b:6f:42:37:27:fb:fc:34:
     9b:5e:e3:dd:53:70:3e:10:17:bb:b2:db:e2:dd:09:aa:ba:56:
     d0:93:c2:b8:bc:7d:4a:d1:d8:51:2c:48:73:7a:af:db:50:58:
     7b:a3:95:62:df:db:46:11:77:b9:0b:21:42:64:ca:e0:a5:e7:
     29:20:07:c5:3e:5e:bf:18:81:28:53:95:89:97:ca:a0:b4:33:
     fa:32:1c:2e:83:a8:86:cf:d0:31:6b:b3:72:eb:f9:6f:3a:ba:
     6a:84:13:e0:57:c7:4d:c2:26:1e:9d:b0:0c:3d:b2:4b:d2:7b:
     80:21:03:72:58:8d:e3:aa:e4:e8:88:db:50:8b:6d:aa:04:3b:
     1a:ba:92:ec:5b:38:52:66:4d:89:08:c3:b3:10:ad:c3:1e:c6:
     50:17:a6:12:58:41:35:d6:6b:ee:e3:29:b7:4e:29:17:2e:6b:
     c9:8a:4a:14:cd:ad:99:88:98:ba:d0:5c:43:64:34:0a:59:5f:
     d4:a5:75:ff:a0:84:8b:8a:f5:35:31:b4:e7:6c:ee:4b:1e:66:
     62:91:fb:90
```

4.2.2 *Online Certificate Status Protocol*

OCSP foi criado como uma alternativa a CRLs, e está especificado no RFC 6960 [3].

Diferentemente de uma lista completa que deve ser obtida pelo cliente periodicamente, com OCSP, o cliente (quem está verificando um certificado) faz uma consulta ao servidor OCSP, perguntando especificamente pelo certificado que quer verificar, através do envio de seu número de série. O servidor (normalmente mantido pela CA), por sua vez, emite uma resposta assinada, indicando se aquele certificado foi revogado ou não.

Entre as vantagens de OCSP estão a não necessidade do cliente ter que manipular listas enormes de certificados revogados. E o fato de que CRLs são atualizadas apenas de tempos em tempos, enquanto que OCSP oferece respostas mais recentes do status de um certificado.

Por outro lado, há a desvantagem de que no processo de validação é preciso fazer uma requisição a mais, e para outro servidor, o que pode aumentar o tempo total necessário para completar a validação. Além disso, essa requisição extra, permite ao servidor OCSP saber quando uma determinada conexão está sendo feita, por quem e para quem, o que é um problema de privacidade.

As duas formas de revogação são seguras, pois tanto a CRL quanto as mensagens OCSP são assinadas digitalmente.

4.3 Autoridades certificadoras

CAs são entidades que assinam e distribuem certificados para terceiros.

O modelo é baseado na confiança depositada nessas autoridades por parte de clientes que estão verificando certificados. Os clientes possuem uma lista local de certificados de CAs nos quais eles confiam.

Por exemplo, navegadores como Firefox e Google Chrome são distribuídos cada um com uma lista de certificados de CAs considerados confiáveis. Ao iniciar conexões HTTPS, o navegador verifica se o certificado apresentado pelo servidor web foi emitido por um dos certificados dessa lista. Se não for o caso, uma mensagem de erro é exibida para o usuário.

4.3.1 Processo de assinatura de um certificado

Normalmente o processo de assinatura acontece da seguinte forma:

O requerente gera um par de chaves localmente.

Em seguida gera uma requisição de assinatura de certificado (CSR, *Certificate Signing Request*, especificada no RFC 2986 [4]). Basicamente ela contém o *Subject* do certificado a ser gerado e a sua chave pública, além de ser assinada pela chave privada do par, para garantir sua integridade.

A CSR então, é enviada para a CA, que verifica se a assinatura e o *subject* estão corretos, gera o certificado e devolve para o requerente.

4.3.2 Cadeias de certificados

O modelo de CAs permite cadeias de certificados.

Cadeias começam com um certificado raiz, que é um certificado de CA auto-assinado. Lembrando que certificados de CA são certificados que podem assinar outros certificados, eles possuem o valor TRUE para o campo CA da extensão *Basic Constraints*.

No processo de verificação de um determinado certificado, primeiro verifica-se se a assinatura desse certificado é válida, em seguida faz-se o mesmo para seu emissor, e assim por diante, emissor por emissor, até se chegar em algum certificado considerado confiável por parte de quem está verificando, que normalmente é um certificado raiz.

CAs podem assinar outros certificados de CA. Isso pode ser usado para delegar a função de assinar certificados para outra organização. Como as principais CAs do mundo são grandes corporações, é comum uma delas assinar um certificado de CA para outra empresa menor, para que ela faça parte da cadeia de certificados comumente considerados confiáveis.

Outra função de cadeias, é permitir que uma CA gere vários certificados de CA para ela mesma, para distribuir a quantidade de certificados assinados. Assim no caso de um deles ser comprometido e precisar ser revogado, isso invalide apenas uma fração do total de certificados assinados por essa organização.

Existe outro campo da extensão *Basic Constraints*, o pathLen. Ele é apenas válido quando o campo CA tem valor TRUE, e a extensão *Key Usage*, contém o valor keyCertSign (que equivale a *Certificate Sign*, quando exibido pelo OpenSSL). Ele serve para indicar o número máximo de certificados intermediários não auto-assinados que podem fazer parte de uma cadeia de validação. Por exemplo, se um certificado de CA tem pahtLen = 2, então entre esse certificado e o certificado que está sendo verificado são permitidos 2 certificados de CA não auto-assinados. Se esse campo não estiver presente, então nenhuma limite é imposto no tamanho da cadeia. É útil para uma CA emitir um certificado de CA que não pode assinar novos certificados de CA, apenas certificados para entidades finais.

4.3.3 Principais problemas de autoridades certificadoras

O grande problema de CAs é que elas são um ponto único de falha em PKI.

É interessante levantar aqui que em segurança da informação não existe um protocolo perfeito ou um modelo perfeito. O modelo mais seguro seria trancar tudo em um cofre e jogar a chave fora, porém isso não faz um sistema

usável. Algum ponto de falha sempre acaba havendo, que não necessariamente precisa ser único, e o problema fica por conta de decidir qual deve ser esse ponto, onde há menos risco, sem destruir a usabilidade.

Basta apenas que uma chave privada de um certificado de CA seja comprometida. Quem obtê-la pode emitir qualquer certificado, e portanto consegue personificar qualquer entidade, assim como realizar ataques de *man-in-the-middle* (para espionar conexões), até o momento que o comprometimento for identificado e esse certificado for revogado, o que pode demorar.

Outro problema existente é quando CAs mal administradas assinam certificados para um determinado domínio sem verificar corretamente se quem fez a requisição é dono desse domínio. Por isso que algumas CAs delegam essa função a uma RA separada.

Capítulo 5

TLS e HTTPS

O objetivo desse capítulo é explicar um pouco do que é TLS e para que serve, sem entrar em muitos detalhes sobre seu funcionamento. Um bom resumo de como ele funciona está disponível em [13].

TLS é um protocolo criptográfico que garante a confidencialidade e integridade da conexão, e também garante autenticação quando certificados digitais são apresentados pelo cliente e/ou servidor.

Sua versão mais recente é a 1.2 e está especificada no RFC 5246 [6]. A versão 1.3 está em processo de desenvolvimento [7].

Anteriormente era conhecido por SSL, *Secure Sockets Layer*, mas as versões mais recentes foram renomeadas para TLS, *Transport Layer Security*. Porém, ainda é comum vê-lo sendo chamada pelo nome antigo, que geralmente é usado como sinônimo de TLS, sem distinção de versão.

TLS é o protocolo padrão para se realizar comunicação segura através de um meio inseguro de comunicação (a internet). É um protocolo bastante robusto, pois diversos ataques contra TLS [8] já foram inventados e até então foi possível mitigar todos eles.

É o TLS que faz autenticação de cliente, a base da solução apresentada nesse trabalho.

5.1 *Handshake*

O *handshake* TLS é o processo que ocorre no ínicio da conexão TLS e serve principalmente para:

- Selecionar a versão do TLS a ser usada.
- Selecionar a *cipher suite* (discutido na próxima seção) que será usada.
- Requisitar e verificar certificados.

Concluído o *handshake*, cliente e servidor podem começar a trocar mensagens, utilizando a *cipher suite* negociada.

5.2 *Cipher suites*

TLS necessita de um conjunto de algoritmos criptográficos para funcionar. Esse conjunto se chama *cipher suite*.

Ao configurar clientes e servidores, é importante selecionar bem quais *ciphers suites* serão permitidas. Pois existem diversas cipher suites que não são mais seguras.

São 4 algoritmos.

5.2.1 Troca de chave

Durante o handshake é feito o processo de troca de chave de sessão.

A chave de sessão trocada serve para fazer a encriptação das mensagens com uma cifra simétrica.

É muito importante utilizar um algoritmo que oferece *forward secrecy*, ou seja, caso a chave privada do servidor (ou do cliente) seja roubada, mensagens trocadas antes do roubo acontecer não poderão ser lidas. Pois o atacante não consegue comprometer a chave de sessão a partir da chave privada.

Um dos algoritmos mais comuns é o ECDHE, uma variação do Diffie-Hellman envolvendo curvas elípticas que é efêmero.

Vale notar que o TLS 1.3 não oferecerá suporte a algoritmos que não oferecem *forward secrecy*.

5.2.2 Autenticação

O algoritmo usado para assinar digitalmente a chave de sessão trocada. Normalmente é o RSA.

5.2.3 Encriptação

As mensagens são encriptadas com uma cifra simétrica, elas são muito mais rápidas do que algoritmos de criptografia de chave pública.

Variações de AES são as mais usadas.

5.2.4 MAC

As mensagens acompanham um MAC. Serve para garantir sua integridade, ou seja, que não foram modificadas no meio do caminho.

MACs baseados em variações de SHA são os mais comuns.

5.3 Autenticação de cliente

A solução apresentada no próximo capítulo funciona com base em autenticação de cliente. Todos os aplicativos móveis deverão apresentar certifica-

dos de cliente na hora de fazer requisições aos serviços do Nubank.

No momento do *handshake* o servidor pode enviar uma mensagem ao cliente requisitando certificados de cliente. O cliente só envia certificados de cliente se o servidor requisitar.

A requisição inclui a lista de certificados que serão usados para validar o certificado de cliente.

Quando o servidor requisita, o cliente precisa obrigatoriamente responder. Se ele não possuir nenhum certificado, deve responder com uma mensagem vazia.

Após recebida a resposta, fica a critério do servidor como continuar a conexão, independente do seu conteúdo.

Normalmente o cliente responde com um certificado somente, e o servidor ao recebê-lo, o verifica e continua apenas se for válido. Ou o cliente não possui nenhum certificado e o servidor decide se continua com a conexão.

Quando ambos o servidor e o cliente apresentam certificados, temos TLS mútuo. No caso em que apenas o servidor apresenta certificado, temos TLS simples.

5.4 HTTPS

HTTPS nada mais é que HTTP sobre TLS. Está especificado no RFC 2818 [9].

Quando uma requisição HTTPS é feita, primeramente um túnel TLS é fechado entre cliente e servidor, e posteriormente as mensagens HTTP são transmitidas por esse túnel.

A diferença é que as URIs começam com "https" em vez de "http", por exemplo `https://www.example.com/~smith/home.html`. E que a porta padrão para o HTTPS é a 443, em vez da 80 do HTTP.

5.4.1 HSTS

HSTS (*HTTP Strict Transport Security*) [10] é uma extensão para o HTTPS que acrescenta um *header* nas respostas HTTPS enviadas pelo servidor. O próprosito desse header é indicar para o cliente que todas a conexões futuras devem ser feitas sempre em HTTPS, nunca em HTTP.

O *header* contém um valor numérico que indica por quanto tempo isso vale (o valor máximo é 1 ano).

Serve para evitar ataques onde um *man-in-the-middle* força o cliente a usar HTTP, que não oferece segurança.

Capítulo 6

A solução desenvolvida

O objetivo do trabalho é tornar mais seguro o acesso dos clientes do Nubank às suas contas, cortando diversas classes de fraudes e minimizando prejuízos aos clientes e à empresa.

Foi buscada uma maneira de restringir operações de maior risco, de forma que elas possam ser efetuadas apenas através do aplicativo para celular, e apenas em um celular autorizado pelo cliente. Isso dificulta diversos ataques possíveis em sistemas não-móveis. Dessa forma qualquer ataque feito por um agente malicioso que depende do acesso direto à conta do cliente passa a ser muito difícil de ser realizado.

A solução desenvolvida para alcançar esses objetivos foi fazer autenticação forte de cliente entre os aplicativos para celular e os serviços do Nubank, utilizando de certificados digitais e TLS mútuo. Assim, não basta a um atacante conhecer o login e a senha de um cliente, ele também precisa dos certificados e chaves privadas do cliente, que nunca saem do celular onde foram geradas, e portanto são difíceis de se obter.

6.1 Visão geral da solução

A arquitetura do Nubank é baseada em microserviços [27] rodando em instâncias EC2 (*Elastic Compute Cloud*) [30] da AWS (Amazon Web Services) [28]. Atualmente são cerca de 60 microserviços, realizando as mais diversas funções. Por exemplo, existe um microserviço dedicado exclusivamente a intermediar autenticação e realizar controle de acesso, outro para manipular o processo de aquisição (o processo onde um possível cliente se torna de fato um cliente), outro para gerar boletos bancários e gerenciar cobranças dos clientes, etc.

A comunicação dos serviços do Nubank com eles mesmos, e entre os aplicativos e serviços acontece através de APIs RESTful usando HTTPS. Os clientes (nesse contexto o software que faz requisições) autenticam os servidores no início da conexão quando fazem uma requisição, pois este apresenta

o seu certificado no *handshake* TLS. Esse certificado é um certificado externo, válido globalmente.

Para alcançar o objetivo de tornar as conexões seguras, queremos também autenticar os clientes. Para isso eles também precisam possuir um certificado para apresentar durante o *handshake*. Sem a apresentação desse certificado, diversas requisições serão bloqueadas pelo servidor.

Os certificados serão emitidos por uma CA própria, ou seja, esses certificados não serão válidos globalmente, apenas serviços e aplicativos móveis do Nubank os aceitarão.

Para os clientes, dois certificados serão emitidos. Um deles será guardado encriptado com a senha do usuário, e só será utilizado para operações mais críticas, onde o cliente precisará digitar sua senha antes de confirmar. O outro não ficará encriptado e será utilizado para todas as outras operações.

Os clientes obterão os certificados ao final do processo de aquisição, ou seja, quando eles se tornam clientes. Nesse momento já é feita uma verificação de identidade e análise de fraude, então é assumido que o certificado está sendo emitido corretamente para o cliente em seu celular.

Clientes atuais do Nubank (ou seja, que já passaram pelo processo de aquisição) e clientes que vierem a perder os certificados (caso ele tenha apagado os dados do aplicativo do Nubank, ou teve seu celular roubado, por exmplo) precisarão passar por um processo de recuperação descrito na seção de vantagens e desvantagens, nesse capítulo.

Além de certificados para clientes, serão emitidos certificados para os funcionários, para que eles acessem os serviços internos da empresa efetuando autenticação de cliente também.

Os certificados serão usados, de maneira automática, para iniciar a comunicação com qualquer serviço do Nubank, assim servindo como uma camada inicial de autenticação e dessa forma mitigando diversos ataques realizados por agentes maliciosos e portanto cortando vários tipos de fraude.

Além disso, a estrutura de autenticação de cliente permitirá futuramente que novas funcionalidades sejam oferecidas aos cliente e funcionários de maneira segura.

6.2 Alternativas buscadas

Antes de se decidir por certificados digitais e TLS mútuo, outras alternativas foram pensadas.

6.2.1 Kerberos

Kerberos é um sistema que funciona baseado em tickets de acesso de vida curta obtidos de um servidor central. Um de seus principais atrativos é que a senha não circula pela rede, outro é que as permissões dadas pelos tickets podem ser bem granulares.

Porém, qualquer solução em Kerberos que vai além de uma rede local tem problema com sincronização de relógio, pois o Kerberos necessita que o relógio esteja sincronizado entre todos os nós da rede, senão os tickets são invalidados. Gerenciar isso é um desafio, pois cada dispositivo precisa estar configurado individualmente.

6.2.2 Desenvolver um protocolo próprio

Foi cogitado o desenvolvimento de um protocolo próprio que consistia na assinatura de requisições, porém a medida em que foi se pensando em todos os problemas e detalhes, essa solução foi se aproximando muito do que é TLS mútuo. E ainda assim haveria o mesmo problema em se distribuir a chave privada que realizaria as assinaturas.

6.2.3 e-CPF (cadastro de pessoa física digital)

O e-CPF [26] é um certificado digital emitido pelo governo federal para pessoas físicas. Ele têm duas versões, a A1 que é apenas um arquivo digital e vale por 1 ano, e a A3 que é um dispositivo de hardware que contém o certificado e vale por 3 anos. O dispositivo é feito para que o arquivo não possa ser extraído facilmente.

O grande atrativo do e-CPF é que ele possui valor legal, documentos assinados por ele não podem ser repudiados. Em vista disso, foi pensado no uso do e-CPF em vez da emissão de certificados próprios. Isso evitaria a necessidade de manter uma CA própria. O resto da solução continuaria a mesma, basicamente.

Porém, o fato de ele custar mais de 160 reais para ser emitido, mais o fato de que apenas o próprio cliente poderia emiti-lo, o que o obrigaria a ter um grande trabalho para instalá-lo no seu celular tornaram essa ideia impraticável.

Finalmente, a solução de certificados digitais e TLS mútuo foi a escolhida, pois apesar de alguns de seus problemas, foi a solução que apresentou o melhor compromisso comparada com as outras.

6.3 Vantagens e desvantagens

A principal vantagem é a exigência do certificado ser apresentado para acessar os serviços, não basta usar a senha. Isso tem o mesmo efeito de autenticação de dois fatores, onde para completar o acesso além de ser necessário usar a senha, algo que se sabe, é necessário usar o certificado, algo que se possui. Dessa maneira, dificulta-se bastante o acesso indevido aos serviços.

Outra grande vantagem é cada cliente e funcionário ter uma chave privada sua, que pode ser utilizada para a transmissão de documentos encriptados e assinados digitalmente. Lembrando que assinaturas garantem integridade, autenticidade e não-repúdio. Isso, potencialmente, pode ser usado em questões legais.

A maior desvantagem da solução se manifesta quando um cliente perde os certificados que já possui, por exemplo no caso de perder ou ter seu celular roubado. Ou simplesmente ter utilizado a função do sistema móvel de voltá-lo para as configurações de fábrica ou a função de apagar os dados do aplicativo do Nubank. Assim apagando o certificado junto.

Quando isso acontecer ele terá de passar por um processo de recuperação para poder emitir novos certificados. Um código será enviado por email ou telefone (através de uma ligação automática, não SMS). O cliente precisará tirar uma foto dele segurando um pedaço de papel com o código anotado e enviar para a equipe do Nubank. A equipe, manualmente, verificará se é mesmo o cliente na foto e se o código está correto, e então o autorizará a obter novos certificados.

Por ser um processo manual, pode demorar um pouco para o cliente receber uma resposta. Porém, decidiu-se que vale a pena este inconveniente, dado que raramente um cliente deve precisar passar por isso.

Eventualmente, novas formas de biometria, que possivelmente utilizam de *machine learning* e envolvem mais automação serão estudadas.

No caso de celular roubado, é esperado que o cliente faça o bloqueio do cartão entrando em contato com o atendimento, ou através do sistema web `https://conta.nubank.com.br`. Sem o bloqueio, o ladrão consegue acessar normalmente a conta do cliente (se conseguir passar pela tela de bloqueio do sistema móvel), da mesma forma que acontece atualmente (sem autenticação de cliente), mas operações importantes ainda exigirão que o usuário digite sua senha.

6.4 A autoridade certificadora

A CA interna criada é composta por um certificado raiz e vários troncos. A raiz, que é auto-assinada, existe apenas para assinar os troncos. Os troncos assinam os certificados finais.

Cada tronco assina certificados para um fim diferente. Há troncos para os certificados de cliente, "customer" e "customer-critical". Há um para os funcionários usarem para acessar os serviços internos, "nubankers". Outro para os funcionários conseguirem conectar na rede da empresa, "network". Entre outros.

Ter um tronco por finalidade ajuda com organização. Pode-se até fazer controle de acesso apenas olhando o tronco do certificado de cliente apresentado (através do campo *issuer*). E também, no caso de comprometimento da

chave privada de um tronco, ao revogá-lo, apenas os certificados assinados por ele serão invalidados.

A chave privada da raiz, após gerada, é usada para assinar o certificado dos troncos. Em seguida é encriptada com um algoritmo de chave simétrica (AES-256-CBC) usando uma chave gerada aleatoriamente. *Secret sharing* é aplicado na chave aleatória, e as partes são distribuídas para pessoas de confiança. Por fim, a chave privada encriptada é armazenada offline. Isso é possível, pois após gerar os troncos, muito raramente a chave privada da raiz será necessária.

Para cada tronco é gerado um arquivo PKCS #12, especificado no RFC 7292 [5]. O PKCS #12 é um formato de arquivo que armazena chaves privadas e certificados. Inicialmente são guardados no S3 (*Simple Storage Service*) [29] da AWS, mas cogita-se futuramente movê-los para o KMS (*Key Management Service*) [32] da AWS, que oferece maneiras mais seguras de se administrar chaves privadas, porém isso ainda precisa ser estudado.

6.5 Processo de assinatura e distribuição de certificados para clientes

Distribuição de certificados é um dos problemas de infraestrutura de chave pública, pois é necessário verificar a identidade de quem está requisitando, para não correr o risco de assinar um certificado para a pessoa errada.

Os clientes obterão seus certificados durante o processo de aquisição, que é o processo através do qual eles se tornam clientes de fato. Nele, o cliente preenche todos seus dados pessoais, incluindo email e telefone, que serão usados no processo de recuperação de certificados (explicado na seção 6.3), caso haja necessidade.

No final do processo, o aplicativo gerará os dois pares de chave e fará uma requisição para o serviço responsável por assinar certificados enviando o número de identificação do cliente (obtido pelo aplicativo em uma etapa anterior) e as duas chaves públicas, além de um JWT (*JSON Web Token*) [11] (obtido no início da aquisição), para garantir que essa requisição está sendo feita pelo mesmo dispositivo que iniciou o processo.

Dessa forma, as chaves privadas do cliente nunca trafegam pela rede, elas nunca saem do dispositivo onde foram geradas.

Esse processo não é perfeito, ainda existem maneiras dele ser explorado. Porém, a aquisição acontece apenas uma vez por cliente, e portanto a superfície de ataque é baixa. Se houve um comprometimento, o cliente verdadeiro não conseguirá acessar sua conta, e entrará em contato com o atendimento. Quando o problema for resolvido, o atacante perderá o acesso à conta do cliente.

6.6 O servidor web e os serviços

O servidor utilizado no Nubank é o Jetty [34]. Cada serviço roda em cima de uma instância do Jetty em uma máquina EC2. É ele que fecha o túnel TLS que comunica com o cliente.

6.6.1 A *keystore* e a *truststore*

Para fechar o túnel, o Jetty precisa de uma *keystore* e uma *truststore*.

A *keystore* é um arquivo que armazena o certificado e a chave privada usada para fechar o túnel.

Lembrando que como esse servidor pode ser acessado também por clientes externos, a *keystore* não pode ter um certificado da CA interna, pois estes não são válidos para tais clientes. É necessário um certificado externo, que para simplificar, pode ser o mesmo para todos os serviços.

A *truststore*, por sua vez, armazena certificados de confiança usados para verificar os certificados de cliente recebidos. Todos os certificados dos troncos e da raiz da CA interna ficam na *truststore*. Apenas os certificados são armazenados, sem suas chaves privadas, pois estas não são necessárias para verificar assinaturas.

A *keystore* e a *truststore* podem ter o formato JKS, o padrão do Java, ou PKCS #12, especificado no RFC 7292 [5]. O escolhido foi o PKCS #12, pois o OpenSSL também consegue manipulá-lo.

6.6.2 Obtendo a *keystore* do S3

O Jetty por padrão lê a *keystore* do disco, e isso apresenta um problema, pois as instâncias são acessíveis pelos desenvolvedores. Se uma conta de desenvolvedor for comprometida fica fácil de roubar a *keystore*.

Por causa disso foi desenvolvida uma classe Java para o Jetty que estende parte do código dele que manipula TLS, sobrescrevendo o método que obtém a *keystore* do disco, por outro que a obtém direto do S3 para a RAM.

Apenas as instâncias dos serviços podem ler a *keystore* do S3, e durante um pequeno intervalo de tempo enquanto ela está subindo.

6.6.3 Configuração do Jetty

Inicialmente a negociação TLS estava a cargo do ELB (*Elastic Load Balancer*) [31] da AWS. O Jetty, que fica atrás do ELB, estava configurado apenas para fazer HTTP.

Portanto, o ELB foi configurado para fazer apenas TCP *passthrough*. E o Jetty foi configurado para fazer TLS, seguindo sua documentação em [35].

As seguintes configurações foram feitas:

- Foram habilitadas as versões 1.0, 1.1 e 1.2 do TLS. A preferência era por manter apenas a versão 1.2, porém, algumas versões antigas do Android não a suportam bem.
- As seguintes *cipher suites* foram habilitadas:
  1. TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  2. TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
  3. TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  4. TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
  5. TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
  6. TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- Renegociação TLS foi desabilitada, pois é vulnerável a ataques.
- HSTS foi ligado.
- Foi habilitada a opção para ser enviada no *handshake* a mensagem pedindo certificados de cliente.

6.6.4 Uma CRL diferente

CRLs, infelizmente, precisam do número de série do certificado. Isso obriga a armazenar todos os números de série no banco de dados. Para evitar esse esforço adicional, foi desenvolvida a seguinte alternativa.

Uma CRL baseada no *subject* do certificado e na data de emissão.

Para implementar isso, foi criado um *handler* [36] para o Jetty que atua após o túnel TLS ser fechado. Ou seja, toda requisição HTTPS feita aos serviços do Nubank, passa pelo *handler* antes de chegar ao serviço.

O *handler* periodicamente atualiza sua cópia local da CRL, obtendo-a do S3. A CRL é um simples arquivo de texto contendo um *subject* e data de revogação por linha.

O *handler*, a cada requisição que chega, pega o certificado de cliente da requisição e verifica se o *subject* daquele certificado está na CRL. Se estiver, então ele verifica a data de revogação. Se o certificado foi emitido antes da data de revogação a requisição é negada e o cliente recebe uma resposta HTTP 401 (não autorizado).

O arquivo está limitado pelo número de *subjects* diferentes que existem. Se um *subject* for revogado uma segunda vez, a entrada antiga é removida, pois se torna desnecessária. Isso diminui o tempo necessário para se fazer uma busca na lista, que foi implementada com a estrutura de dados ConcurrentHashMap do Java [37].

Apenas um serviço vai escrever nesse arquivo do S3, evitando problemas de concorrência na atualização da CRL.

6.6.5 Serviços

Os serviços no Nubank normalmente são feitos em Clojure [39] e usam o Pedestal [40], um *framework* web.

Cada um deles roda em um contêiner que já possui o Jetty configurado.

O Jetty está configurado para rejeitar conexões com certificados inválidos ou revogados. Porém, ele não pode bloquear conexões sem certificado, pois existem rotas nos serviços que devem ser acessíveis mesmo sem certificado de cliente.

Nos serviços, as rotas precisam ser configuradas ou como públicas (aceitam acesso sem certificado) ou como privadas (exigem um certificado de cliente). Se o serviço possuir alguma rota sem nenhuma das duas marcações, ele não consegue subir. Isso foi feito para se evitar que alguma rota privada seja esquecida de ser marcada, o que é uma falha de segurança.

Além disso, com o Pedestal, é possível extrair o certificado a partir do objeto da requisição. As rotas que exigem certificado fazem isso, e verificam as informações dele para decidir se a requisição deve ser permitida ou não.

6.7 Autenticação de cliente entre serviços

Além de clientes e funcionários, há outro agente para utilizar certificados de cliente. Os serviços.

Os serviços normalmente existem para receber requisições, e portanto atuam como servidor no modelo cliente-servidor. Porém, muitas vezes um serviço precisa se comunicar com outro. Para tal, ele faz requisições HTTPS ao outro serviço, atuando então como cliente no modelo cliente-servidor.

Nesse momento, os serviços também realizarão autenticação de cliente. Cada um terá um certificado próprio. Esse certificado é diferente do certificado usado como servidor.

Esses certificados serão gerados através da execução de uma função no AWS Lambda [33], de maneira que quem dispara a execução nunca tem acesso à chave privada gerada. O arquivo PKCS #12 contendo a chave privada e o certificado assinado serão guardados no S3, e seu acesso só será permitido às instâncias EC2 dos respectivos serviços.

6.8 Certificados de cliente e os aplicativos móveis

Um dos objetivos da solução é afetar o mínimo possível a usabilidade do cliente.

Portanto, os aplicativos serão modificados para gerenciar e usar os certificados da maneira mais automática possível. Sem que os mecanismos sejam expostos para o usuário e ele possa continuar utilizando o aplicativo como já o faz.

O único momento afetado é quando o usuário for acessar o aplicativo a partir de outro dispositivo que não possui certificados, nesse caso ele precisará passar pelo processo de recuperação para emitir novos.

6.9 Possibilidades futuras

O ganho de segurança dado por autenticação de cliente mais o fato do cliente passar a ter uma chave privada sua certamente oferecem possibilidades de criação de novas funcionalidades e elaboração de novos mecanismos.

Existem dois em particular que serão comentados aqui.

Primeiro, cartões de crédito virtuais. Cartões virtuais são números de cartão de crédito efêmeros que são gerados para serem utilizados apenas em um único serviço online. A vantagem dele, é que caso o número do cartão seja armazenado e posteriormente exposto, ele não terá validade e qualquer compra com ele será bloqueada. Isso é um grande ganho de segurança que evita muitas fraudes, o que é bom tanto para o cliente, quanto para o emissor do cartão.

É desejado implementá-los direto no aplicativo, através dele o cliente conseguirá gerá-los. Sem a solução de autenticação de cliente, bastaria um atacante descobrir a senha do cliente, que ele poderia gerar quantos cartões quisesse e fazer muitas compras em seu nome. O risco é tão grande que torna essa funcionalidade impraticável. Mas com autenticação de cliente, apenas o dispositivo já autorizado poderá gerar cartões virtuais, tornando essa funcionalidade viável.

Outra possibilidade, mais ambiciosa, é a implementação de uma solução inspirada em BeyondCorp [38], do Google.

Normalmente redes empresariais baseiam sua segurança na camada de rede, controlando vários acessos através de um *firewall*. Mas na prática, hoje em dia, é muito comum os serviços internos serem acessados remotamente por dispositivos móveis e notebooks, normalmente através de uma VPN (*Virtual Private Network*). Esse modelo de controle de acesso é muito vulnerável, pois basta quebrar a barreira inicial que se tem acesso a tudo.

BeyondCorp busca uma maneira de resolver esse problema. Ao invés de se bloquear os serviços na camada de rede, todos ficam disponíveis na internet, e o controle de acesso é feito baseado no dispositivo que está sendo usado e em credenciais de acesso. Ou seja, há uma camada de alto nível que faz um controle mais fino de autenticação e autorização para os serviços internos.

Certificados de cliente, podem ser usados para fazer a autenticação dos dispositivos, servindo como uma base que viabiliza a implementação dessa ideia.

Capítulo 7

Conclusão

Neste trabalho foi apresentada uma solução de segurança da informação baseada em autenticação de cliente realizada através de TLS e certificados digitais emitidos por uma autoridade certificadora interna.

Foi mostrado que a solução alcança o objetivo proposto de restringir a realização de determinadas operações feitas através de requisições HTTPS apenas a dispositivos que contenham certificados de cliente para serem apresentados no momento de início das requisições. Dessa forma corta-se por completo diversas classes de fraudes e alcança-se um grande aumento no nível de segurança para todos os serviços oferecidos pela organização.

A solução foi descrita com o objetivo de que outras organizações possam se inspirar nela e adaptá-la a seus casos particulares. Em especial, foi mostrada uma maneira de realizar o processo de distribuição de certificados, onde se encontra o maior inconveniente da solução.

Foi mostrada que a solução pode servir como uma alternativa a autenticação de dois fatores, com o benefício extra de fornecer um par de chaves de criptografia de chave pública que pode ser utilizada para diversas operações que envolvam encriptação e assinatura digital de dados e mensagens. E também, que ela serve tanto para aumentar a segurança para os serviços oferecidos aos clientes, quanto para os serviços utilizados internamente por funcionários.

O custo de desenvolvimento, implementação e manutenção da solução é alto, mas os benefícios de segurança ganhos são muito grandes e portanto a solução compensa para organizações que possuem necessidade crítica de segurança na realização de suas operações.

Referências Bibliográficas

[1] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
https://tools.ietf.org/html/rfc5280

[2] A String Representation of Distinguished Names (X.500)
https://tools.ietf.org/html/rfc1779

[3] X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP
https://tools.ietf.org/html/rfc6960

[4] PKCS #10: Certification Request Syntax Specification Version 1.7
https://tools.ietf.org/html/rfc2986

[5] PKCS #12: Personal Information Exchange Syntax v1.1
https://tools.ietf.org/html/rfc7292

[6] The Transport Layer Security (TLS) Protocol Version 1.2
https://tools.ietf.org/html/rfc5246

[7] The Transport Layer Security (TLS) Protocol Version 1.3 draft-ietf-tls-tls13-07
https://tools.ietf.org/html/draft-ietf-tls-tls13-18

[8] Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)
https://tools.ietf.org/html/rfc7457

[9] HTTP Over TLS
https://tools.ietf.org/html/rfc2818

[10] HTTP Strict Transport Security (HSTS)
https://tools.ietf.org/html/rfc6797

[11] JSON Web Token (JWT)
https://tools.ietf.org/html/rfc7519

[12] Abstract Syntax Notation One (ASN.1)
http://www.itu.int/itu-t/recommendations/rec.aspx?rec=x.680

[13] TLS summary
http://security.stackexchange.com/a/20847

[14] Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf

[15] Secure Hashing
http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html

[16] Shamir's Secret Sharing Scheme
http://point-at-infinity.org/ssss

[17] Credential Stealing as an Attack Vector
https://www.schneier.com/blog/archives/2016/05/credential_stea.html

[18] Secrecy, Security, and Obscurity
https://www.schneier.com/crypto-gram/archives/2002/0515.html#1

[19] Security vs Usability
https://www.schneier.com/blog/archives/2009/08/security_vs_usa.html

[20] Security Design: Stop Trying to Fix the User
https://www.schneier.com/blog/archives/2016/10/security_design.html

[21] The Unfalsifiability of Security Claims
https://www.microsoft.com/en-us/research/wp-content/uploads/2015/09/unfalsifiabilityOfSecurityClaims.pdf

[22] Cross-site Scripting (XSS)
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[23] Cross-Site Request Forgery (CSRF)
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

[24] The Heartbleed Bug
http://heartbleed.com

[25] Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems
https://42xtjqm0qj0382ac91ye9exr-wpengine.netdna-ssl.com/wp-content/uploads/2015/08/TimingAttacks.pdf

[26] e-CPF: Certificado Digital para Pessoa Física
https://serasa.certificadodigital.com.br/p/e-cpf

[27] Microservices
http://martinfowler.com/articles/microservices.html

[28] Amazon Web Services (AWS)
https://aws.amazon.com

[29] AWS Simple Storage Service (S3)
https://aws.amazon.com/s3

[30] AWS Elastic Compute Cloud (EC2)
https://aws.amazon.com/ec2

[31] AWS Elastic Load Balancer (ELB)
https://aws.amazon.com/elasticloadbalancing

[32] AWS Key Management Service (KMS)
https://aws.amazon.com/kms

[33] AWS Lambda
https://aws.amazon.com/lambda

[34] Jetty
https://www.eclipse.org/jetty

[35] Jetty: Configuring SSL/TLS
https://www.eclipse.org/jetty/documentation/9.3.x/configuring-ssl.html

[36] Jetty: Handlers
https://www.eclipse.org/jetty/documentation/9.3.x/jetty-handlers.html

[37] ConcurrentHashMap
https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.html

[38] BeyondCorp: A New Approach to Enterprise Security
https://www.usenix.org/system/files/login/articles/login_dec14_02_ward.pdf

[39] Clojure
http://clojure.org

[40] Pedestal
http://pedestal.io