UNIVERSIDADE FEDERAL DE GOIÁS – REGIONAL CATALÃO
DEPARTAMENTO IBIOTEC
PROJETO FINAL DE CURSO EM CIÊNCIA DA COMPUTAÇÃO

AVALIAÇÃO DE SEGURANÇA DO USO DE TLS EM ESTAÇÕES SERVIDORAS NA REDE DA UNIVERSIDADE FEDERAL DE GOIÁS

Lorena Aparecida Rezende

MONOGRAFIA

CATALÃO – GO, 2016

LORENA APARECIDA REZENDE

AVALIAÇÃO DE SEGURANÇA DO USO DE TLS EM ESTAÇÕES SERVIDORAS NA REDE DA UNIVERSIDADE FEDERAL DE GOIÁS

Monografia apresentada como requisito parcial para a obtenção do diploma de Bacharel em Ciência da Computação pela Universidade Federal de Goiás – Regional Catalão.

Orientador:
Ricardo Couto Antunes da Rocha

CATALÃO – GO

2016

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Aparecida Rezende, Lorena
Avaliação de Segurança do Uso de TLS em Estações Servidoras na Rede da Universidade Federal de Goiás [manuscrito] / Lorena Aparecida Rezende. - 2016.
101 f.: il.

Orientador: Prof. Ricardo Couto Antunes da Rocha .
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Unidade Acadêmica Especial de Biotecnologia, Ciência da Computação, Catalão, 2016.
Bibliografia. Anexos.
Inclui símbolos, tabelas, lista de figuras, lista de tabelas.

1. HTTPS. 2. Servidores. 3. SSL. 4. TLS. 5. UFG. I. , Ricardo Couto Antunes da Rocha, orient. II. Título.

CDU 004

LORENA APARECIDA REZENDE

AVALIAÇÃO DE SEGURANÇA DO USO DE TLS EM ESTAÇÕES SERVIDORAS NA REDE DA UNIVERSIDADE FEDERAL DE GOIÁS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal de Goiás – Regional Catalão.

Trabalho aprovado em 16 de agosto de 2016.

Ricardo Couto Antunes da Rocha
Orientador

Márcio de Souza Dias
Universidade Federal de Goiás

Tércio Alberto dos Santos Filho
Universidade Federal de Goiás

Catalão – GO

2016

Aos meus pais, José Francisco e Aparecida Nunes, irmão, Thiago, que sempre foram a base de tudo. Aos meus amigos, que sempre me motivaram e auxiliaram quando precisei. Aos professores que tive, por terem sido verdadeiros mestres, exemplos de profissionais e amigos. Eu amo vocês!

Agradecimentos

Em primeiro lugar quero agradecer à Deus porque é graças à Ele que consegui chegar onde cheguei, devo à Ele tudo que fiz e tudo que sou. Sei que muitas das vezes as Suas mãos tomaram as minhas para que eu pudesse continuar à escrever este trabalho e que são estendidas sobre minha fronte, todos os dias, para me abençoar e acalentar meu coração. Quero agradecer, em especial, aos meus pais e irmão, que acreditaram em mim, mesmo quando nem eu mais acreditava, eles que sofreram comigo à cada dificuldade que encontrei, eles que também vibraram de felicidade com cada conquista minha, à eles que relevaram o meu mau humor nos dias ruins. Quero agradecê-los porque souberam ser FAMÍLIA todas as vezes que precisei. Quero agradecer aos meus amigos Edmar, Diego, Cleriston e Yulle, pelo laço de amizade que ultrapassa a distância e o tempo, agradecer pelas confidências e incentivo constante para que eu pudesse continuar e me graduar, mesmo que por meio de apostas hahaha. Agradeço à Anelisa, ao Welliton (vulgo 3 ranchos), ao Iago e aos Gustavos da minha vida, pelo carinho, pelas risadas, pelas cervejas e piadas, pela cumplicidade e polêmicas no café da manhã (algoritmo da mochila, realização de estudos para cura do Alzheimer). Agradeço à Jéssica que (sempre vou dizer) é minha guia espiritual, que sempre zela e intercede ao Pai por mim, ela que sempre me cuidou e cuida (eu te amo, flor!). Quero agradecer ao meu orientador, Ricardo, pela paciência e orientação, se não fosse pela ajuda, eu jamais conseguiria concluir o trabalho. Quero agradecer também ao Vaston, pela amizade, pela cumplicidade, pelo carinho e respeito que se estendeu para além das salas de aula. À Erika só tenho à agredecer e "*the little things* - Colbie". Bom, os nomes são muitos e eu peço desculpas por não conseguir citar todos (viu? Lazinha, Iury, Jorge, Célio, Bayara, Soneca, Melque, Caio, amigos do trabalho), mas quero agradecer à todas as pessoas que fazem parte da minha vida, porque cada um, de maneira particular, agregou o melhor para a minha vida. O meu sentimento de gratidão por tudo e por todos é tão grande que mal consigo me expressar, me perdoem! Então, por enquanto, obrigada! Obrigada por fazerem eu me sentir uma das pessoas mais felizes e maravilhadas nessa vida. Eu amo muito cada um de vocês, muito!

“Só sei que nada sei” (Sócrates)

Resumo

REZENDE, L. A.. *Avaliação de Segurança do Uso de TLS em Estações Servidoras na Rede da Universidade Federal de Goiás*. 2016. 101 f. Monografia – Departamento IBIOTEC, Universidade Federal de Goiás – Regional Catalão, Catalão – GO.

O SSL (*Secure Socket Layer*), ou TLS (*Transport Layer Security*), surgiu em 1994 e se tornou um padrão de comunicação que utiliza a criptografia, assinaturas e certificados digitais para garantir a segurança ponto-a-ponto, à nível de transporte. Se opera com o protocolo de aplicação HTTP, por exemplo, obtemos um serviço de entrega seguro fornecido pelo protocolo conhecido como HTTPS. Em 2010, na conferência *Black Hat*, Ivan Ristic apresentou o *Internet SSL Survey* que consiste na análise da segurança de todos os servidores TLS da Internet com base na validade do certificado, tamanho de chaves, algoritmos de assinatura, nomes de domínio, versões do protocolo, cifras, algoritmos de troca de chaves e renegociação de configuração. Este trabalho é baseado em sua apresentação, à fim de avaliar a segurança do uso do TLS em estações servidoras na rede da Universidade Federal de Goiás (UFG) para verificar quais servidores são seguros ou não, bem como, identificar possíveis ataques. Neste trabalho 66 servidores TLS na rede da UFG foram analisados e classificados como seguros ou inseguros de acordo com o tamanho da chave privada, algoritmo de *hash*, validade do certificado e versões do protocolo que são utilizadas. Para analisar os parâmetros de configurações foram utilizadas as ferramentas *ssllabs-scan* e *ssl-lufg*. A *ssllabs-scan* é uma ferramenta *open source* disponibilizada pela *Qualys*, que simula o protocolo *handshake* e armazena em um arquivo (*.json*) todas as configurações que o servidor TLS aceita. A *ssl-lufg* é uma ferramenta que foi desenvolvida para o auxílio das verificações, consiste em obter dos arquivos *.json* as informações referentes aos quatro parâmetros avaliados e, inferir quais servidores são seguros ou inseguros e o porquê. Após a análise realizada foi constatado que, embora a diferença seja pequena, a maioria dos servidores utilizam o TLS seguro e que três servidores possivelmente sofreram algum ataque, mas estes não são discutidos neste trabalho.

Palavras-chaves: Assinaturas Digitais, Certificados Digitais, Criptografia, HTTPS, Servidores, SSL, TLS, UFG.

Lista de ilustrações

Lista de quadros

Sumário

CAPÍTULO 1

Introdução

Geralmente o transporte das informações é baseada no protocolo TCP (*Transmission Control Protocol*) e oferece um serviço de entrega confiável para a camada de aplicação. Este serviço envia as mensagens e verifica se elas foram recebidas pelo destinatário, mas não realiza nenhum serviço de autenticação e não utiliza nenhum tipo de criptografia para proteger os dados durante a transmissão, de maneira que esta pode estar vulnerável à diversos tipos de ataques (FOROUZAN B.A.; FEGAN, 2008). Caso exista um atacante entre as partes que se comunicam, este não será identificado, podendo então ter acesso às mensagens sigilosas e até mesmo alterar totalmente o seu conteúdo (RISTIĆ, 2014a).

Em virtude do aumento dos serviços fornecidos pela *web* aumentou também os acessos e a preocupação com a segurança das informações que são enviadas. Para tanto, em 1994, a Netscape desenvolveu um protocolo baseado no protocolo TCP à fim de garantir a segurança à nível de transporte em uma comunicação ponto-a-ponto. Este protocolo foi denominado como SSL (*Secure Socket Layer*), que deu origem ao TLS (*Transport Layer Security*) e se tornou um padrão de segurança na Internet (STALLINGS, 2008).

A garantia de segurança do TLS é obtida por meio do uso da criptografia, assinaturas e certificados digitais, fazendo com que as mensagens sejam íntegras, autênticas e lidas apenas por pessoas autorizadas. É muito importante ter cuidado quanto à esolha dos parâmetros de configuração do TLS, pois se não estiver configurado corretamente, cria-se apenas uma ilusão de que está sendo oferecido um serviço seguro (RISTIĆ, 2014b).

Um servidor que possui ao menos um parâmetro de configuração inseguro pode estar vulnerável à uma diversidade de ataques e o acesso não autorizado à informações sigilosas pode ocasionar danos irreparáveis, como prejuízos à empresa ou à reputação de pessoas (STALLINGS, 2008).

1.1 Objetivos

Baseado na análise feita por Ivan Ristić [1] sobre o uso seguro do TLS, que verifica a segurança de todos os servidores TLS da Internet, este trabalho realiza a análise da segurança dos servidores TLS na rede da Universidade Federal de Goiás (UFG). O objetivo é verificar quais servidores estão configurados de maneira insegura e informar aos seus responsáveis, para que possam ter conhecimento e tomar as providências necessárias à fim de torná-los seguros, evitando assim a ocorrência de ataques bem sucedidos. Para tal, foi realizado uma varredura na rede da UFG para identificarmos quais servidores aceitam conexão TLS e como resultado, foram encontrados 66 deles.

A análise dos 66 servidores TLS é realizada com base nos parâmetros de configuração referentes ao tamanho da chave privada, algoritmo de *hash* utilizado para as assinaturas, validade do certificado X.509 e as versões do protocolo TLS que o servidor oferece suporte. Para analisar os parâmetros de configurações foram utilizadas as ferramentas *ssllabs-scan* e *ssl-lufg*.

A *ssllabs-scan* é uma ferramenta *open source* disponibilizada pela *Qualys*, que simula o protocolo *handshake* e armazena em um arquivo (*.json*) todas as configurações que o servidor TLS aceita. A ferramenta *ssl-lufg* foi desenvolvida ao longo deste trabalho para auxiliar nas verificações e inferências, que consiste em obter dos arquivos *.json* as informações referentes aos quatro parâmetros avaliados e, determinar quais servidores são seguros ou inseguros e o porquê. O resultado da análise realizada pela *ssl-lufg* é armazenado em um arquivo *.txt*, que está disponível no Anexo 2. Além da análise dos quatro parâmetros de configuração, foi verificado também os nomes de domínios presente nos certificados digitais, dos quais se houver alguma anomalia é possível concluir que houve algum ataque, mas estes não foram verificados e nem tratados neste trabalho.

1.2 Organização do trabalho

Esta monografia está organizada da seguinte maneira: o Capítulo 2 apresenta os fundamentos de segurança, dentre eles: criptografia simétrica, assimétrica e híbrida, assinatura digital, autenticação com chave pública e certificado digital. O Capítulo 3 apresenta o protocolo TLS, bem como o seu modelo de funcionamento baseado nos subprotocolos *handshake*, de registro e alertas. O Capítulo 4 apresenta os parâmetros de segurança do TLS e também os ataques mais conhecidos. O Capítulo 5 apresenta a análise do uso do TLS em estações servidoras na rede da Universidade Federal de Goiás. E, por fim, as conclusões são apresentadas no Capítulo 6.

[1] RISTIĆ, I. Internet SSL Survey 2010. Black Hat Technical Security Conference. 2010. Las Vegas, EUA. Disponível em <https://media.blackhat.com/bh-us-10/presentations/Ristic/BlackHat-USA-2010-Ristic-Qualys-SSL-Survey-HTTP-Rating-Guide-slides.pdf>

CAPÍTULO 2

Fundamentos de Segurança

O protocolo TLS tem como objetivo garantir a segurança à nível de transporte em uma comunicação ponto-a-ponto (STALLINGS, 2008). E, para garantir a segurança é necessário levar em consideração quatro conceitos fundamentais: autenticação, não repúdio, integridade e confidencialidade (TRINTA; MACEDO, 1998), estes que são definidos conforme segue:

- Autenticação: é a prova de que o autor da referida mensagem é, de fato, quem a enviou (STALLINGS, 2008).
- Não repúdio: garantia de que se um autor criou e enviou uma mensagem, este não pode negá-la (TANENBAUM, 2002).
- Integridade: assegura que mensagem é íntegra, ou seja, não foi modificada por terceiros, de maneira intencional ou não (STALLINGS, 2008).
- Confidencialidade: que as informações contidas no documento não sejam lidas por outros que não estejam autorizados (STALLINGS, 2008).

Para atender à esses requisitos de segurança, o protocolo TLS utiliza a criptografia, assinaturas e certificados digitais, estes que são discutidos neste capítulo.

Estes conceitos são discutidos neste capítulo.

2.1 Criptografia

De origem grega, criptografia, vem de *kryptos* = escondido/oculto e *grifo* = grafia. É o mecanismo adotado para garantir a segurança das informações protegendo-as do acesso não autorizado. A criptografia garante a confidencialidade por meio de algoritmos públicos, com funções matemáticas, que são responsáveis por tornar as mensagens ilegíveis às pes-

soas não autorizadas, ou seja, a mensagem só será legível para a(s) pessoa(s) autorizada(s), portanto o destinatário deve ter a chave correta para decifrar e ler a mensagem.

Para que as mensagens se tornem ilegíveis é necessário que sejam cifradas por completo, *bit*-a-*bit*, em um fluxo contínuo, ou divididas em blocos de *n*-bytes e criptografados em seguida. Estes processos de cifragem são denominados como *criptografia de fluxo de dados* e *criptografia de blocos*, respectivamente (MORENO *et al.*, 2005).

A criptografia das mensagens pode ocorrer por meio de *códigos* ou *cifras*. Ao utilizar a criptografia por códigos uma palavra, ou conjunto delas, é subtituída por outras equivalentes. Sua utilização não é muito viável pois são fáceis de serem descobertos e só se pode enviar mensagens predefinidas (TRINTA; MACEDO, 1998). Na criptografia por *cifras* as letras são embaralhadas ou substituidas por meio de algoritmos, estas são mais difíceis de serem decifradas e podem ser utilizadas para enviar *n* mensagens. O processo de cifragem pode ocorrer de duas maneiras: por *transposição* ou *substituição*. Nas cifras de *transposição* as letras são embaralhadas e nas cifras de *substituição* cada letra, ou um conjunto delas, são substituídas por outras de acordo com uma tabela. Há quatro tipos de cifras de substituição:

- *Cifras de Substituição Simples*: consiste em uma tabela usada para a cifragem das mensagens, na cifra de César, por exemplo, utiliza-se a 3ª letra do alfabeto após a original. Exemplo: A = D.
- *Cifras de Substituição Polialfabética*: consiste na utilização de várias cifras de substituição simples, com diferentes valores.
- *Cifras de Substituição de Poligramas*: utiliza-se um grupo de caracteres, em vez de um, para substituir. Exemplo: "ABA"= MAE, "ABB"= JKI.
- *Cifras por Deslocamento*: não utiliza um valor fixo para o deslocamento, cada letra é substituída por um critério de rotação e se for necessário o critério pode ser repetido. Exemplo: carro; critério: 023, a informação dada pelo critério é de que devemos substituir a primeira letra por 0, a segunda pela segunda letra do alfabeto que está à sua frente, etc.

Nas cifras de substituição é chamado de *chave* o valor relacionado ao deslocamento das letras à frente, na cifra de César, por exemplo, o valor da chave é 3, pois é a quantidade de letras à frente da letra original. Utilizar uma chave com tamanho 3 significa que há $2^3 = 8$ possíveis valores para a chave, dessa maneira, quanto maior a chave, maior o custo computacional para tentar descobrí-la (TRINTA; MACEDO, 1998).

Conforme mencionado, os algoritmos criptográficos são públicos e consequentemente avaliados por diversos criptoanalistas que buscam por vulnerabilidades no código, dessa forma, não se deve utilizar algoritmos que são considerados vulneráveis, pois em al-

gum momento teria-se uma brecha para quebrar o código (MORENO *et al.*, 2005). As tentativas de violar um sistema seguro são chamadas de *ataques*, que tem por objetivo quebrar uma mensagem criptografada decifrando-a sem ter o conhecimento da chave utilizada. Um exemplo é o ataque pela *força bruta*, que consiste em utilizar todas as chaves, uma em seguida da outra, para tentar decifrar a mensagem (STALLINGS, 2008).

Embora seja recomendável utilizar chaves com tamanho considerado seguro também é necessário dar atenção à sua origem, ou seja, sobre quais condições ela foi gerada. Uma *chave* é um conjunto de números, ou caracteres alfanuméricos, gerados de maneira aleatória e não repetíveis. O algoritmo para tal é conhecido como *Geradores de Números Pseudo-Aleatórios* (*GNPAs*), que deve utilizar entradas diferentes para cada vez que for executado de maneira que o resultado final também será diferente (números não repetíveis). Não adiantaria ter uma chave com tamanho grande se o conjunto de entrada utilizado pelo *GNPAs* for sempre o mesma, isto acabaria gerando chaves repetidas e portanto, facilitaria sua descoberta (MORENO *et al.*, 2005) (NAKAMURA; GEUS, 2007).

Baseado no uso das chaves a criptografia é classificada em três tipos: de *chave simétrica*, *assimétrica* e *híbrida* (NAKAMURA; GEUS, 2007).

2.1.1 Criptografia Simétrica

Na criptografia simétrica, ou de chave privada, tanto o emissor quanto o receptor utilizam a mesma chave para cifrar/decifrar a mensagem, dessa maneira, existe uma chave para cada par de pessoas que queiram se comunicar. Esta é exatamente uma das desvantagens de se utilizar a criptografia de chave simétrica, pois pode tornar-se inviável quando se tratando de um grupo muito grande, visto que haveria a necessidade de gerenciar várias chaves e distribuí-las, em um meio inseguro, para cada par de pessoas (TRINTA; MACEDO, 1998).

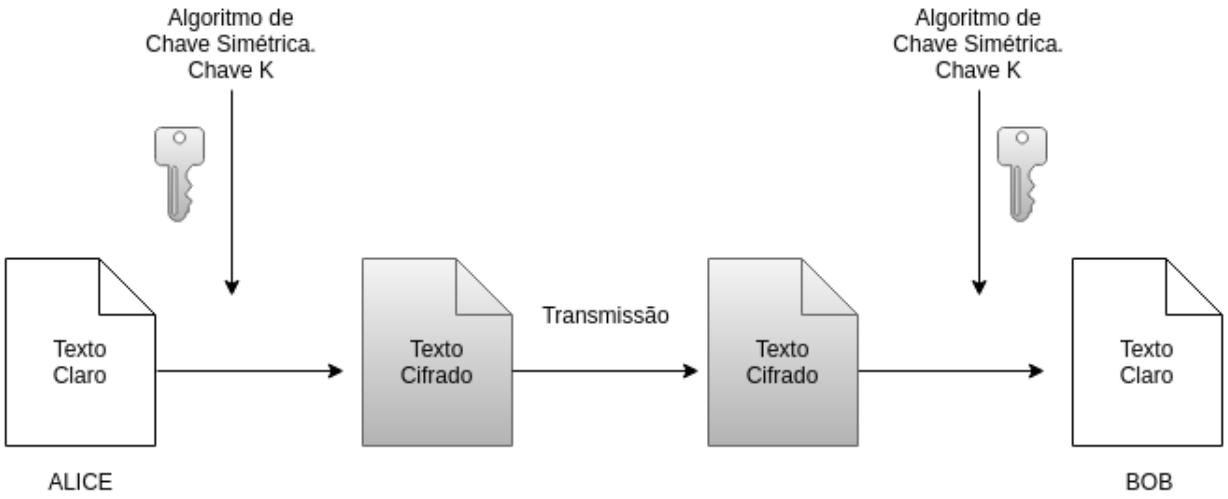
Para tentar solucionar o problema de distribuição de chaves, foi desenvolvido o centro de distribuição de chaves (*KDC*, do inglês *Key Distribution Center*), que possui o conhecimento das chaves utilizadas pelos usuários confiados à este. Neste caso, um usuário *A* envia ao *KDC* uma mensagem cifrada com a sua chave (chave de *A*). O *KDC* decifra a mensagem com a chave de *A*, cifra com chave do remetente, no caso é a chave de *B* e, envia ao destinatário: o usuário *B*, que por fim, decifra a mensagem utilizando a sua chave (TRINTA; MACEDO, 1998). O processo de cifrar/decifrar utilizando algoritmo de chave simétrica é representado pela Equação (1):

$$C = E_k(P), então P = D_k(C) \tag{1}$$

onde, P é o texto claro (mensagem legível), C é o texto claro P criptografado pela função E, com a chave K. E, D é função de decifrar, que utiliza a mesma chave K, usada para cifrar, para obter a mensagem original P. A Figura 1 ilustra o exemplo do uso de criptografia de chave

simétrica, na qual *ALICE* cifra uma mensagem com a chave secreta *K* e envia a mensagem cifrada para *BOB*. Ao receber a mensagem de *ALICE*, *BOB* utiliza a mesma chave para decifrar a mensagem e ter acesso ao texto claro (legível).

Figura 1 – Representação do modelo de criptografia simétrica.



Fonte: Autoria própria

Este processo é mais rápido se comparado à criptografia de chave assimétrica, mas continua sendo inseguro devido à distribuição centralizada, fazendo com que o *KDC* se torne alvo de ataques (TRINTA; MACEDO, 1998). São exemplos de algoritmos que utilizam chave simétrica: DES, Triple-DES, Rijndael, RC2, RC4, AES, IDEA e Skipjack (MORENO *et al.*, 2005).

2.1.2 Criptografia Assimétrica

A criptografia de chave assimétrica, ou de chave pública, minimiza o problema de distribuição de chaves encontrado na criptografia de chave simétrica, pois não é preciso estabelecer um canal seguro para tal, uma vez que as mensagens são cifradas por meio da chave pública do remente. Esta abordagem oferece um serviço de autenticação porque utiliza um par de chaves, totalmente diferentes, para cifrar/decifrar uma mensagem. A chave pública é utilizada para cifrar a mensagem e a chave privada para decifrar (NASCIMENTO, 2009).

Uma chave pública é publicada em um diretório público, permitindo que qualquer pessoa possa enviar mensagens cifradas (com a chave pública do remetente), as quais somente o proprietário da chave privada, referente ao par da chave pública utilizada, é capaz de decifrá-las (autenticação). É muito importante manter a chave privada em sigilo, do contrário, qualquer pessoa que tenha acesso à esta chave poderia decifrar as mensagens criptografadas e realizar a leitura das mesmas, quebrando totalmente um dos princípios que

garantem a segurança: o sigilo (NAKAMURA; GEUS, 2007). Os processos para cifrar/decifrar é expresso conforme as Equações (2) e (3), respectivamente:

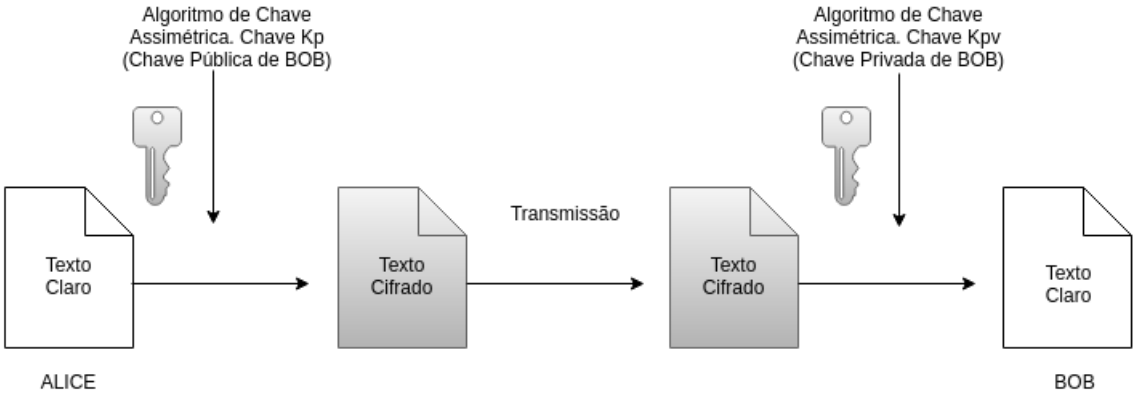
$$C = E_{Kp}(P) \tag{2}$$

$$P = D_{Kpv}(C) \tag{3}$$

onde, P é o texto claro, C é a mensagem P criptografada pela função E, que utiliza a chave pública Kp do remetente. E, D é a função de decifrar C (que é mensagem P criptografada pela função E), que utiliza a chave privada Kpv para obter a mensagem original P (TRINTA; MACEDO, 1998).

Levando em consideração o cenário descrito, caso um usuário (*ALICE*) queira enviar mensagens à um outro usuário (*BOB*) é necessário que *ALICE* realize a criptografia de um texto claro (mensagem legível) utilizando a chave pública *Kp* que foi disponibilizada por *BOB* e a envie para este. Ao receber a mensagem enviada por *ALICE*, *BOB* utiliza sua chave privada *Kpv* para decifrar a mensagem e ter acesso ao texto claro (NASCIMENTO, 2009). Este exemplo é representado na Figura 2.

Figura 2 – Representação do modelo de criptografia assimétrica.



Fonte: Autoria própria

Alguns exemplos de algoritmos de chave assimétrica são: Diffie-Hellman, RSA e Merkle-Hellman, este último baseia-se no algoritmo do problema da mochila (TANENBAUM, 2002).

2.1.3 Criptografia Híbrida

A criptografia híbrida é a combinação da criptografia simétrica e assimétrica à fim de utilizar as vantagens de cada uma delas: processo de cifragem rápido, serviço eficiente na distribuição de chaves e suporte à assinatura digital. Consiste em utilizar algoritmos de chave pública para realizar a autenticação, estabelecer um canal seguro e então, poder enviar a chave secreta à ser utilizada na encriptação das mensagens. Por exemplo, uma pessoa

A deseja se comunicar com uma pessoa *B* e *B* possui uma chave pública, então *A* gera uma chave privada (à ser utilizada para cifrar as mensagens que serão trocadas) e a criptografa com a chave pública de *B*. Quando *B* receber a mensagem, se for realmente o proprietário do par de chave pública utilizado por *A*, poderá decifrá-la e ter acesso à chave secreta que *A* enviou. Ao decifrar a mensagem com a sua chave privada, *B* é autenticado e ambos passam a ter o conhecimento da chave secreta à ser utilizada, logo, poderão cifrar as mensagens com esta chave. Esta abordagem é utilizada pelos protocolos TLS, SET e IPSec, pelo programa de criptografia PGP, S/MIME, e a especificação X.509. Quando combinados esses dois tipos de criptografia podemos ter os serviços de confidencialidade e assinatura digital, este que será discutido posteriormente (MORENO *et al.*, 2005).

2.2 Assinaturas Digitais

A assinatura digital é o mecanismo utilizado provar se o autor de um determinado documento é quem diz ser e que o documento não foi modificado/violado por terceiros de maneira acidental ou intencional. Seu objetivo não é garantir a confidencialidade, mas a autenticidade e a integridade das mensagens. Para atender à essas garantias a assinatura digital utiliza a criptografia de chave assimétrica e uma função de *hash*, para que a assinatura possa ser checada por qualquer pessoa que tenha conhecimento da chave pública do remetente e que seja possível verificar a integridade das mensagens (KUROSE; ROSS, 2010).

Uma função de *hash* é vista como uma função de resumo que utiliza como entrada uma mensagem, de tamanho qualquer, para criar um bloco de dados de tamanho fixo que pode ser verificado, mas não recuperado. Este bloco de dados é utilizado para permitir a verificação da integridade de uma mensagem, ou seja, ao executar uma função de *hash* em uma mensagem, será produzido um *hash* de tamanho fixo, que é enviado ao destinatário junto com a mensagem original. Quando o destinatário receber a mensagem e aplicar nela a mesma função de *hash* utilizada pelo emissor, deverá obter um *hash* idêntico ao recebido, pois do contrário, significa que a mensagem foi modificada (KUROSE; ROSS, 2010) (NAKAMURA; GEUS, 2007).

Embora função de *hash* permita que a integridade da mensagem seja verificada, este não possui nenhum tipo de autenticação, por exemplo, *ALICE* está trocando mensagens com *BOB*, mas *MARIA* (que é uma terceira pessoa) gera uma mensagem, calcula o *hash*, anexa-o à mensagem e envia para *BOB*. Quando *BOB* receber a mensagem irá aplicar a mesma função de *hash* (utilizada pelo emissor) e se obter um *hash* idêntico ao recebido, significa que a mensagem é íntegra, mas não é possível verificar se a mensagem recebida foi enviada pela *ALICE* ou por outra pessoa. Dessa forma, é necessário utilizar o *hash* com uma chave de autenticação (KUROSE; ROSS, 2010).

As assinaturas digitais utilizam a função de *hash* e a criptografia assimétrica, como

chave de autenticação, para oferecer o serviço de autenticação e verificação de integridade da mensagem. Conforme pode ser visto na Figura 3, para assinar e verificar um documento digitalmente é necessário:

1. Aplicar uma função de *hash* no documento original, o que resultará em um bloco de *hash*.

2. Criptografar, com a chave privada do emissor, o *hash* originado da função anterior anexá-lo ao documento. O *hash* é criptografado para que o receptor possa verificar a autenticidade da mensagem recebida.

3. Quando o documento é recebido, o receptor utiliza a chave pública do remetente para decifrar a assinatura que está anexada ao documento, neste momento, o emissor é autenticado pelo receptor, pois somente o portador da chave privada (par da chave pública utilizada) seria capaz de ter cifrado o *hash* em questão.

4. Em seguida, aplica-se a mesma função de *hash*, utilizada pelo emissor, no documento recebido, o que resultará em um outro *hash*.

5. O receptor, por fim, compara estes dois últimos *hashes*. Eles devem ser idênticos, do contrário significa que o conteúdo da mensagem foi alterado, mas não se sabe onde ou quando foi modificado (NAKAMURA; GEUS, 2007).

   Exemplos de algoritmos de *hash* são: MD4, MD5, SHA1, SHA256 (MORENO *et al.*, 2005).

2.2.1 MAC

Um código de autenticação de mensagem, ou MAC (do inglês, *Message Authentication Code*), também tem por objetivo garantir a autenticidade e a integridade da mensagem, mas diferente das assinaturas digitais, um MAC é gerado e verificado por meio de uma chave secreta, portanto não oferece o serviço de assinatura (TRINTA; MACEDO, 1998).

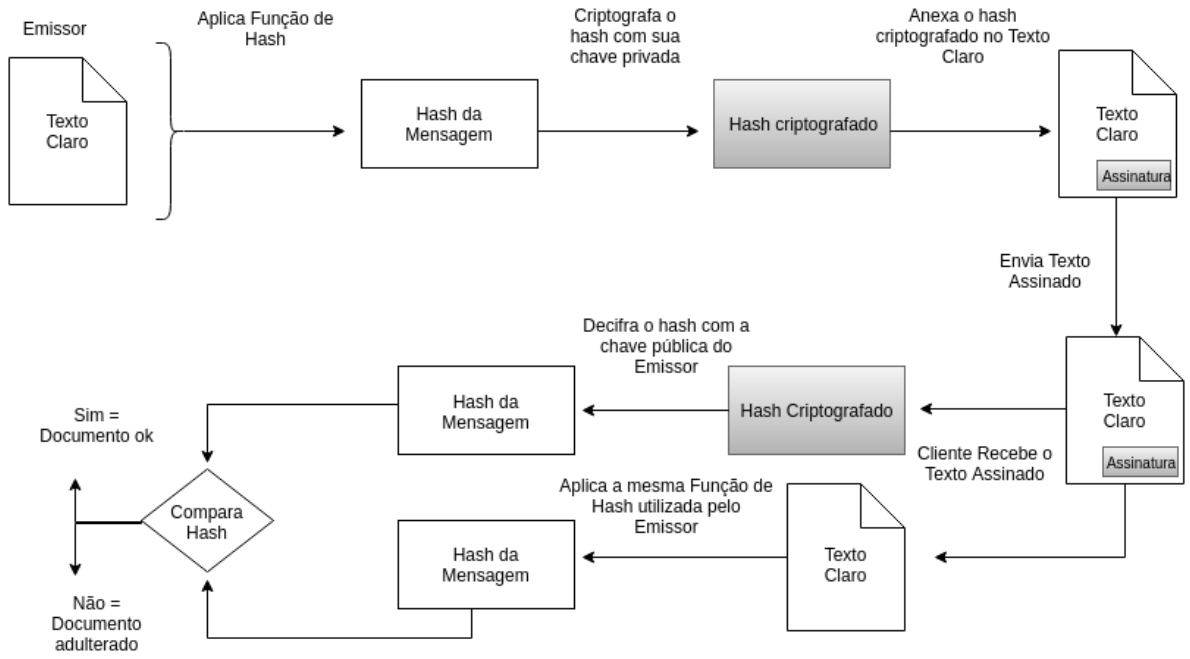
O código de autenticação é gerado por meio de uma chave secreta e um algoritmo MAC aplicado em uma mensagem de tamanho arbitrário, conforme pode ser expresso na Equação (4).

$$MAC = F(M, C) \tag{4}$$

onde, o MAC é gerado por meio de uma função F aplicada em uma mensagem M utilizando uma chave secreta C.

Quando alguém recebe a mensagem com o código de autenticação poderá verificá-lo ao aplicar a mesma função MAC utilizada pelo emissor, com a chave secreta compartilhada entre eles. Após calcular MAC, o código de autenticação gerado é comparado ao recebido e

Figura 3 – Representação do esquema de assinatura e verificação da assinatura.



Fonte: Autoria própria

se forem idênticos, significa que a mensagem não foi alterada e é autêntica, pois apenas ambas as partes envolvidas possuem o conhecimento da chave secreta utilizada (STALLINGS, 2008). A Figura 4 apresenta a sequência de passos para gerar e verificar um código de autenticação de mensagem.

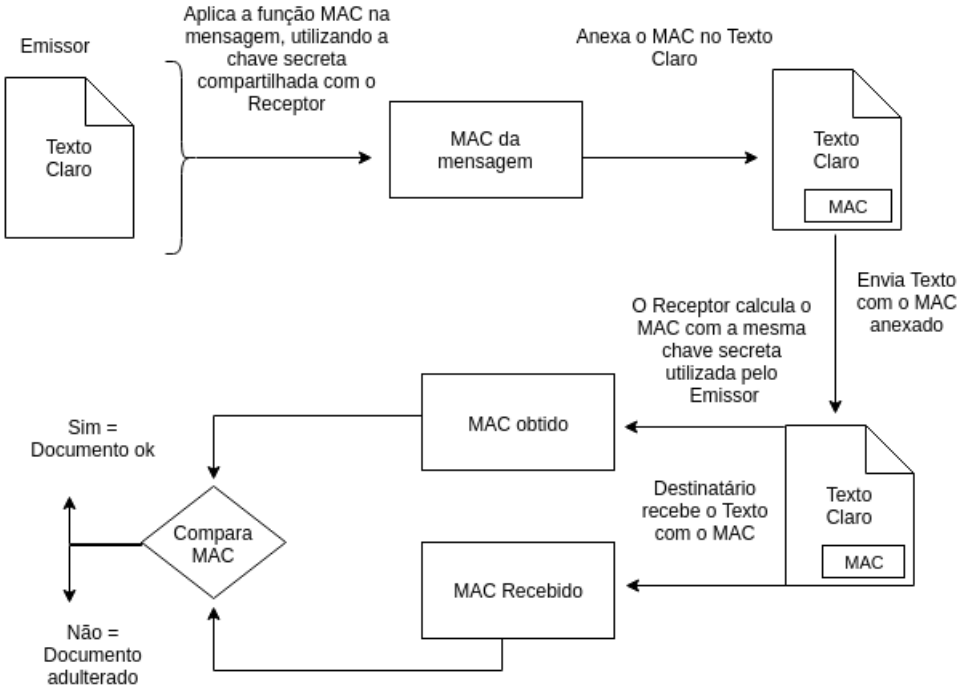
2.3 Autenticação com Chave Pública

A autenticação com chave pública consiste no uso de assinaturas digitais para realizar a autenticação de uma parte, ou ambas, que queiram estabelecer uma conexão. Para que as assinaturas sejam autenticadas, as chaves públicas precisam ser distribuídas, de maneira segura, para que possam chegar àqueles que desejam se comunicar. Uma maneira de distribuir essa chave pública é utilizando certificados digitais, que utilizam cadeias de confiança para validar a chave pública que lhe diz respeito (SANTIN *et al.*, 2012). Na próxima sessão os certificados digitais serão discutidas com mais detalhes.

2.4 Certificados Digitais

Os certificados digitais são portadores de chave pública, utilizados para identificar seu proprietário fazendo a associação de um nome com uma chave pública e são validadados por meio da assinatura de uma Autoridade Certificadora (AC). No certificado está pre-

Figura 4 – Representação do uso de código de autenticação de mensagens (MAC).



Fonte: Autoria própria

sente informações do seu proprietário, sua chave pública, período de validade e a assinatura da AC que foi a responsável por verificar a veracidade das informações e validar todos os dados dispostos no certificado (NASCIMENTO, 2009).

O proprietário de um certificado deve ter um par de chaves: púbica e privada que podem ser geradas pelo si próprio ou emitida por uma AC. A chave pública fica disposta no certificado (chave pública do servidor), enquanto a chave privada deve ficar com o proprietário e em segurança, pois caso outra pessoa tenha acesso, poderá se passar pelo proprietário legítimo.

Se um atacante desconhece a chave privada do proprietário legítimo, poderá obter um par de chaves por sua conta e anexar uma nova chave pública no certificado, esta que será diferente do par utilizado pelo proprietário legítimo. Quando o cliente utilizar a chave pública da AC que o assinou para validar a assinatura de quem o emitiu, irá verificar que a chave presente no certificado não corresponde à chave que a AC utilizou para assinar o certificado. Ou seja, o cliente irá utilizar a chave pública da AC, presente no repositório de ACs válidas, para decifrar o *hash* da assinatura, ao decifrar o *hash* e compará-lo com o *hash* obtido pela aplicação da função (de *hash*) no certificado, irá identificar que os dois são diferentes, de maneira que não é possível validar o certificado.

No momento que o cliente solicita a assinatura do certificado digital, assina um documento com a sua chave privada, no qual prova para a AC que é realmente o portador

daquele determinado par de chaves, assim a AC é capaz de vincular a chave privada ao proprietário e assina o certificado com a sua chave privada. O certificado também pode ser auto-assinado, o que não é viável, à menos que este seja reconhecido como uma autoridade certificadora.

Todo certificado é emitido e assinado pela autoridade certificadora que o criou. Uma AC é responsável por gerenciar certificados: emitir, revogar, armazenar, renovar e distribuir, de acordo com leis, políticas e regras utilizadas por uma AC chamada de AC raiz. Não existe apenas uma AC, até porquê com o número crescente de usuários solicitando certificados, apenas uma AC não conseguiria atender à demanda e, como há várias autoridades certificadoras atuando, há a necessidade de gerenciá-las, assim surgiu a *PKI* (*Public-Key Infraestructure*), conhecida como infraestrutura de chave pública. Esta é uma infraestrutura de pessoas, hardware e softwares, que trabalham com leis, políticas e regras que são utilizadas para além de criar, revogar, armazenar, distribuir certificados, atualizar o par de chaves utilizados (quando expirados ou caso sejam comprometida) e fiscalizar as ACs subsequentes (NASCIMENTO, 2009).

Uma autoridade certificadora deve ter credibilidade de confiança, uma boa reputação e prezar pelo elevado grau de segurança, além de ser conhecida pelos *browsers*, para que estes possam verificar a validade do certificado quando um usuário tentar acessar algum sítio, para tanto, existe uma cadeia hierárquica para que as ACs possam ser validadas. A verificação das assinaturas param em uma AC raiz, ou seja, à quem deu autorização e reconhecimento à todas as outras ACs que estão atuando. A AC raiz também é a responsável por atualizar a lista de certificados válidos e revogados, e expedir certificados para outras ACs (DIERKS; RESCORLA, 2008).

No Brasil a *PKI* é conhecida como ICP-Brasil, constituída em agosto de 2001 (BURNETT; PAINE, 2002) (NASCIMENTO, 2009) (NAKAMURA; GEUS, 2007). São exemplos de autoridades certificadoras no Brasil: Caixa Econômica Federal (AC-CAIXA), Autoridade Certificadora da Justiça (AC-JUS), Serasa Experian (AC-SERASA), dentre outras (DIERKS; RESCORLA, 2008). Para a distribuição de chaves públicas na *web* é utilizado o padrão de certificado X.509 (NASCIMENTO, 2009), que será discutido na próxima sessão.

2.4.1 Padrão de Certificado X.509

Baseado num modelo de assinaturas digitais e criptografia de chave pública, o modelo X.509 surgiu em 1988, como um padrão de certificado digital, que vincula um nome à uma chave pública e tem por objetivo fornecer serviços de autenticação. Vários protocolos utilizam este modelo de autenticação, dentre eles: IPSec, SET, S/MIME e o TLS, este último que é o foco deste trabalho. Os itens que seguem estão presentes num certificado X.509 são:

- *Versão do padrão X.509* que está sendo utilizada;

- *Número de série*: número inteiro emitido pela AC à fim de associá-lo ao certificado;
- *Nome do emissor*: nome de quem criou e assinou o certificado (AC);
- *Período de validade do certificado*: datas com formato "*Not Before*: Mês Dia Horas Ano Fuso Horário"e "*Not After*: Mês Dia Horas Ano Fuso Horário";
- *Nome do proprietário*: referente ao proprietário legítimo do certificado;
- *A chave pública do proprietário*;
- *ID exclusivo do emissor*: utilizado para identificar, exclusivamente, a entidade emissora (AC);
- *ID exclusivo do proprietário*: utilizado para identificar, exclusivamente, o proprietário legítimo do certificado;
- *Assinatura da AC*: um *hash* que tem como entrada todas as informações supracitadas, criptografadas com a chave privada da AC emissora;
- *Algoritmo de assinatura*: disponibiliza a informação de qual o algoritmo de *hash* utilizado pela assinatura;

A Figuras 5 apresenta o modelo de certificado X.509, no qual podemos identificar todos os campos supracitados. O servidor TLS utilizado para exemplo foi o *ead.inf.ufg.br*. No exemplo da Figura 5 é possível identificar a versão do padrão X.509 utilizada (versão 3), o número serial atribuído pela AC, que o algoritmo utilizado para a assinatura digital é o SHA1, com o algoritmo de criptografia RSA. As informações do emissor são: País(*C*): Brasil, Estado(*ST*): Goiás, Cidade(*L*): Goiânia, Órgão Emissor(*O*): Universidade Federal de Goiás, Departamento(*OU*): Instituto de informatica, Nome Comum/identificador único(*CN*): *ead.inf-.ufg.br*. O período de validade é de: 22/08/2013 à 21/08/22/08/2016.
"*Subject*"e "*Issuer*"são parâmetros que descrevem os dados únicos do proprietário do certificado e da autoridade certificadora, respectivamente, dessa forma, para este exemplo, nota-se que os parâmetros do emissor(*Issuer*) e do proprietário(*Subject*) são idênticos, podendo concluir que o certificado é auto assinado.

No certificado X.509v3 existe um campo CA = "?", que é utilizado para representar se é aceitável algum tipo de extensão: informação de chaves e políticas, restrições de certificação e, informações de emissor e proprietário, mas estas não serão discutidas, apenas que se esta opção está como TRUE (CA=TRUE) e o *browser* não a reconhece, envia uma mensagem de certificado inválido (STALLINGS, 2008).

Não é recomendado assinar seus próprios certificados, pois no momento da sua validação poderá acontecer de o *browser* consultar o repositório de AC válidas e não reconhecer a assinatura, de maneira que não consegue validá-la e então, apresentará a informação de

Figura 5 – Padrão de certificado X.509, utilizado pelo servidor *ead.inf.ufg.br*.

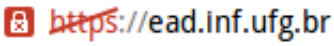
```
lorena@lorenarezende: ~
lorena@lorenarezende:~$ openssl x509 -in ead.inf.ufg.br -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            97:c5:20:1b:1c:4d:38:b6
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=BR, ST=Goias, L=Goiania, O=Universidade Federal de Goias, OU=Instituto de Informatica, CN=ead.inf.ufg.br
        Validity
            Not Before: Aug 22 12:56:18 2013 GMT
            Not After : Aug 21 12:56:18 2016 GMT
        Subject: C=BR, ST=Goias, L=Goiania, O=Universidade Federal de Goias, OU=Instituto de Informatica, CN=ead.inf.ufg.br
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:be:d9:9d:88:cd:da:a9:ca:73:67:44:77:e1:53:
                    98:47:f0:7a:90:12:ac:50:a0:e9:92:35:6d:f4:aa:
                    2e:6f:ef:9d:aa:e9:48:d6:ac:a7:29:f7:42:38:f3:
                    ff:7b:46:47:94:aa:62:54:cc:1e:4a:f3:45:54:6e:
                    95:60:cf:00:2c:51:b9:1e:8e:18:93:ec:8d:e7:ae:
                    f9:e2:84:2e:54:01:9c:99:04:e7:f7:97:ac:de:05:
                    0d:89:87:43:ef:f2:14:bc:76:14:61:9b:9e:6c:f0:
                    bd:1c:68:33:83:ee:b2:f7:9c:c4:d8:c8:ca:cf:52:
                    1b:a5:5f:15:92:09:ea:fe:9d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                44:DF:AD:59:AD:70:FF:83:A1:53:DE:DE:41:9B:FA:66:AA:AC:77:1E
            X509v3 Authority Key Identifier:
                keyid:44:DF:AD:59:AD:70:FF:83:A1:53:DE:DE:41:9B:FA:66:AA:AC:77:1E

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha1WithRSAEncryption
         af:a4:6d:b2:9d:6c:9b:51:f9:d0:46:0f:18:3a:c9:35:54:cd:
         96:a5:b5:18:3f:f5:91:77:ce:d6:a0:d7:52:5f:d3:78:1a:4f:
         a3:b6:5d:15:66:c6:fd:31:b4:85:21:2a:87:ff:69:80:22:e1:
         04:87:9c:81:cc:48:e1:0c:9a:f1:17:47:60:86:1c:42:ec:56:
         a3:69:35:0f:bc:3e:cd:92:83:69:95:6d:07:22:eb:df:75:83:
         2a:1d:a8:d3:42:a1:34:ea:93:31:e1:d4:8b:05:8b:a1:5f:7d:
         43:45:ff:90:47:83:3b:60:19:b5:79:94:58:e8:f9:8d:2a:e8:
         19:47
-----BEGIN CERTIFICATE-----
MIIC9jCCAl+gAwIBAgIJAJfFIBscTTi2MA0GCSqGSIb3DQEBBQUAMIGTMQswCQYD
VQQGEwJCUjEOMAwGA1UECAwFR29pYXMxEDAOBgNVBAcMB0dvaWFuaWExJjAkBgNV
BAoMHVVuaXZlcnNpZGFkZSBGZWRlcmFsIGRlIEdvaWFzMSEwHwYDVQQLDBhJbnN0
aXR1dG8gZGUgSW5mb3JtYXRpY2ExFzAVBgNVBAMMDmVhZC5pbmYudWZnLmJyMB4X
DTEzMDgyMjEyNTYxOFoXDTE2MDgyMTEyNTYxOFowgZMxCzAJBgNVBAYTAkJSMQ4w
DAYDVQQIDAVHb2lhczEQMA4GA1UEBwwHR29pYW5pYTEmMCQGA1UECgwdVW5pdmVy
c2lkYWRlIEZlZGVyYWwgZGUgR29pYXMxITAfBgNVBAsMGEluc3RpdHV0byBkZSBJ
bmZvcm1hdGljYTEXMBUGA1UEAwwOZWFkLmluZi51ZmcuYnIwgZ8wDQYJKoZIhvcN
AQEBBQADgY0AMIGJAoGBAL7ZnYjN2qnKc2dEd+FTmEfwepASrFCg6ZI1bfSqLm/v
narpSNaspyn3Qjjz/3tGR5SqYlTMHkrzRVRulWDPACxRuR6OGJPsjeeu+eKELlQB
nJkE5/eXrN4FDYmHQ+/yFLx2FGGbnmzwvRxoM4PusvecxNjIys9SG6VfFZIJ6v6d
AgMBAAGjUDBOMB0GA1UdDgQWBBRE361ZrXD/g6FT3t5Bm/pmqqx3HjAfBgNVHSME
GDAWgBRE361ZrXD/g6FT3t5Bm/pmqqx3HjAMBgNVHRMEBTADAQH/MA0GCSqGSIb3
DQEBBQUAA4GBAK+kbbKdbJtR+dBGDxg6yTVUzZaltRg/9ZF3ztag11Jf03gaT6O2
XRVmxv0xtIUhKof/aYAi4QSHnIHMSOEMmvEXR2CGHELsVqNpNQ+8Ps2Sg2mVbQci
699lgyodqNNCoTTqkzHh1IsFi6FffUNF/5BHgztgGbV5lFjo+Y0q6BlH
-----END CERTIFICATE-----
lorena@lorenarezende:~$
```

Fonte: Terminal Linux

que a conexão não é segura. Conforme apresentado na Figura 6, é emitido pelo *browser* uma mensagem de alerta ao acessar o sítio *ead.inf.ufg.br*, informando que a conexão não é segura. Ao clicar no cadeado com o "X"vermelho e, em seguida, em "detalhes", é possível verificar que o servidor não é seguro porque seu certificado não é assinado por uma autoridade certificadora confiável, conforme está ilustrado na Figura 7.

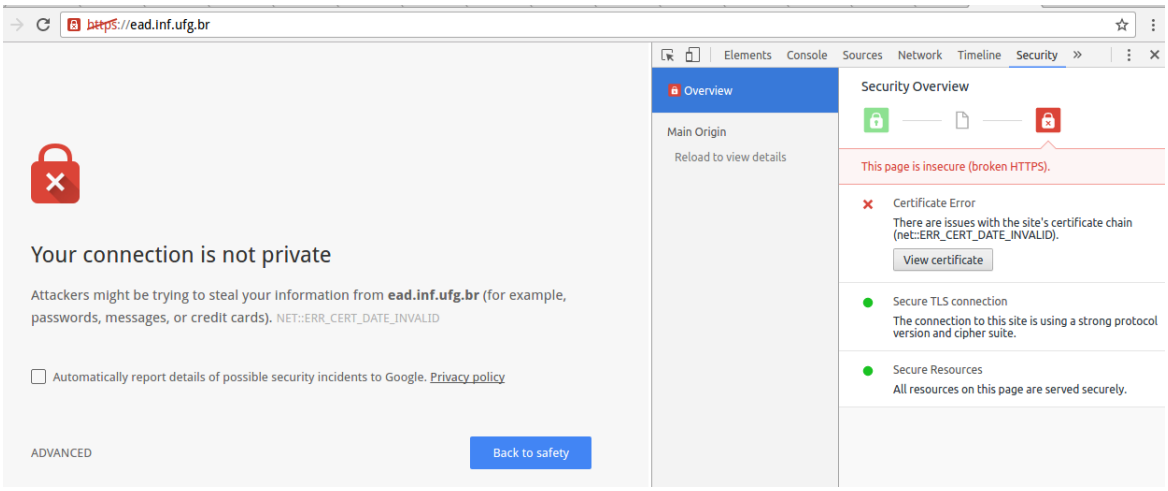
Ao clicar em "*View certificate*"é possível ter acesso ao certificado digital. Na Figura 8 está o certificado referente ao servidor *ead.inf.ufg.br* que foi utilizado como exemplo. Observa-se que os parâmetros descritos em *Issued To* (proprietário do certificado) é idêntico aos des-

Figura 6 – "X"na conexão via https ao tentar acessar o servidor *ead.inf.ufg.br*.



Fonte: Endereço *ead.inf.ufg.br* via navegador Google-Chrome.

Figura 7 – Mensagem de certificado inválido.

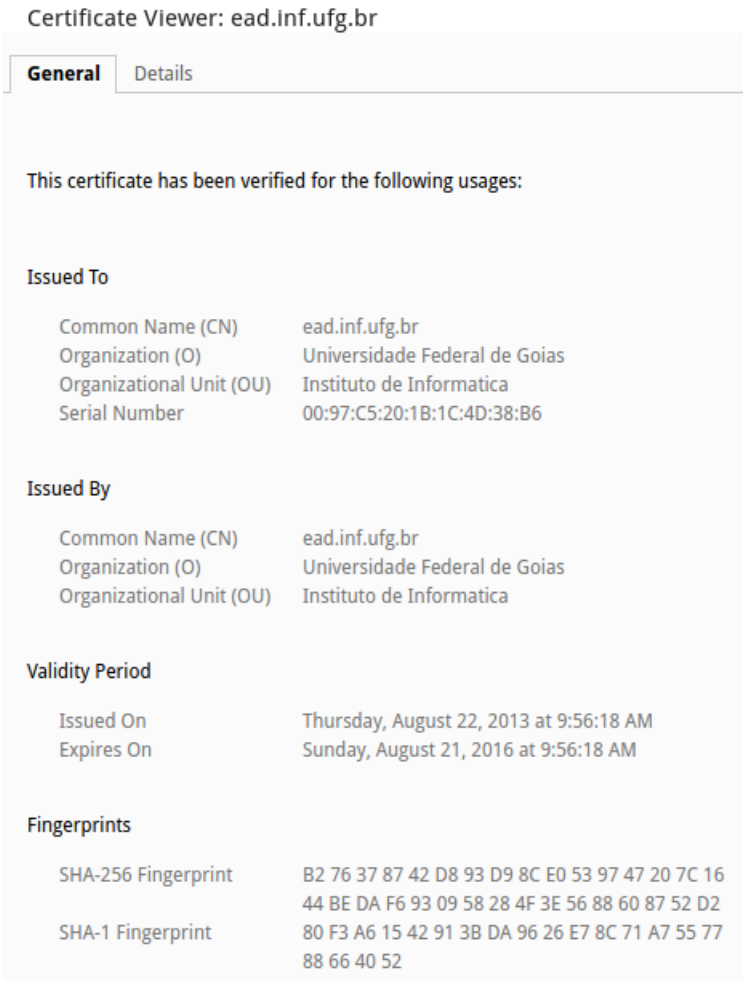


Fonte: Navegador Google-Chrome

critos em *Issued By* (emissor do certificado), de maneira que este certificado é auto assinado e que o emissor não é uma autoridade certificadora reconhecida.

Com base na fundamentação dos conceitos de criptografia, assinaturas e certificados digitais, será discutido no Capítulo 3 o funcionamento do TLS, à fim de demonstrar o uso destes conceitos na garantia da autenticação do servidor, integridade, condifencialidade e autenticidade das mensagens enviadas.

Figura 8 – Exibição do certificado pelo navegador.



Fonte: Navegador Google-Chrome

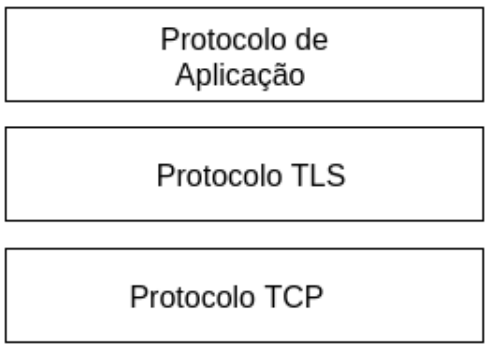
CAPÍTULO 3

Protocolo TLS

O protocolo SSL (*Secure Socket Layer*), ou TLS (*Transport Layer Security*), surgiu em novembro de 1994, pela Netscape, à fim de garantir a privacidade, autenticidade e integridade dos dados transmitidos em uma comunicação ponto-a-ponto. Este protocolo é baseado no protocolo de transporte TCP e se tornou um padrão de segurança na comunicação *web* (STALLINGS, 2008). É embutido nos navegadores e utiliza assinatura e certificado digital, criptografia simétrica, assimétrica e infraestrutura de chaves públicas, conhecida como *PKI* (*Public-Key Infraestructure*), para oferecer serviços de autenticação do servidor, confidencialidade, integridade e autenticidade das mensagens enviadas.

O protocolo TLS oferece um serviço de entrega segura para os protocolos de aplicação, geralmente opera na porta 443 e é utilizado com o protocolo de transporte TCP à fim de oferecer um serviço de transmissão seguro. Quando um cliente envia uma requisição de acesso HTTPS à um site, a requisição HTTP é enviada sob a conexão TLS usando a porta de conexão 443 (STALLINGS, 2008). A Figura 9 ilustra o uso do TLS dentro do modelo de camadas de rede do TCP/IP.

Figura 9 – Representação do modelo de uso do TLS.



Fonte: Autoria própria

A primeira versão do TLS (SSLv2) foi implementada em março de 1994, no navegador da Netscape 1.1, mas devido à vulnerabilidades descobertas, no final do mesmo ano, a em-

presa implementou o SSLv3, que foi um padrão de segurança até 1999. Em janeiro de 1999 foi lançado o TLS1.0, assim denominado por questões políticas entre a Microsoft e Netscape. Em abril de 2006 liberaram a versão TLS1.1 e, em agosto de 2008 a versão TLS1.2, que é a mais atual e portanto, a mais segura (RISTIĆ, 1999) (RISTIĆ, 2014a).

3.1 Funcionamento

Para entender o funcionamento do protocolo TLS é necessário entender os conceitos de conexão e sessão. A *conexão* TLS é o transporte de uma informação ponto-a-ponto e cada uma está relacionada à uma sessão. Uma *sessão* é a associação entre cliente e servidor, com os respectivos parâmetros de segurança compartilhados por várias conexões. Cada conexão está associada à uma sessão e uma sessão pode estar associada à n conexões. Se os parâmetros de segurança não fossem especificados na sessão, seria necessário negociá-los para cada conexão, causando uma sobrecarga no funcionamento do protocolo (STALLINGS, 2008).

O protocolo TLS estabelece dois subprotocolos principais: o protocolo *handshake* e o protocolo de *registro* (STALLINGS, 2008).

3.1.1 Protocolo *Handshake*

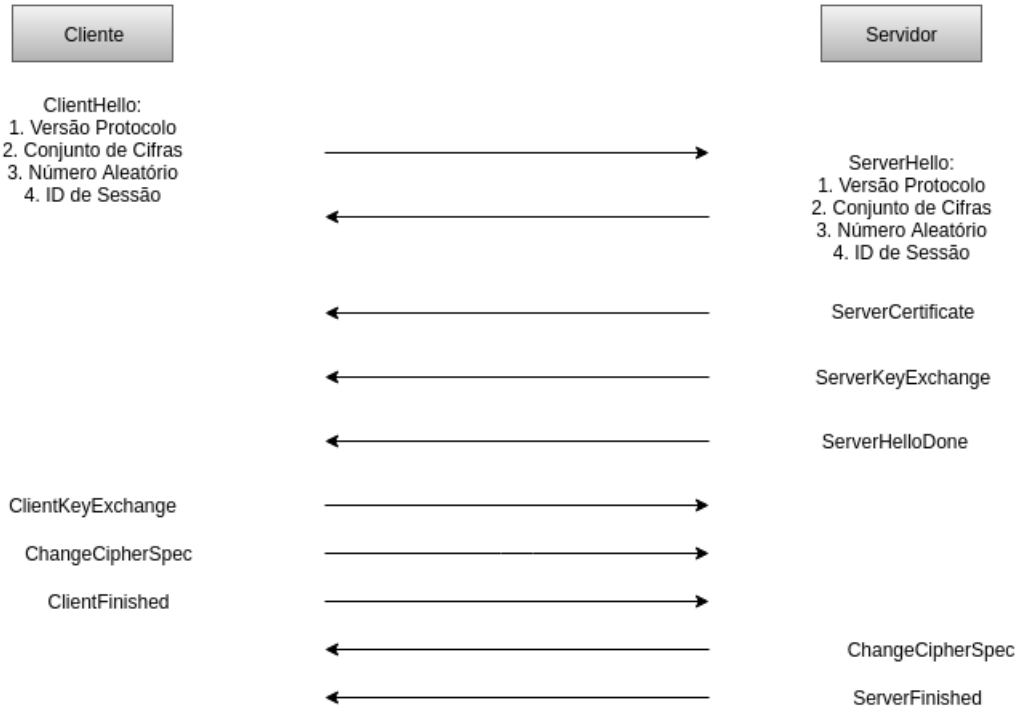
O *handshake*, ou protocolo de estabelecimento de sessão, é a etapa inicial para estabelecer uma conexão TLS. É o responsável por realizar a autenticação do servidor e negociar os parâmetros de segurança que serão utilizados na conexão. Neste processo pode também ser verificada a autenticidade do cliente, mas este serviço é opcional. Conforme visto na Figura 10, durante o *handshake*, os seguintes tipos de mensagens são trocadas entre um servidor e um cliente (RISTIĆ, 2014b):

1. *ClientHello* - Nesta mensagem o cliente envia uma requisição de conexão para o servidor, na qual possui seus parâmetros de configurações para que o servidor possa negociar o conjunto de cifras, versão do protocolo, um número aleatório, um ID de sessão e algoritmos de compressão.

2. *ServerHello* - O servidor negocia, de acordo com os parâmetros enviados pelo cliente, e envia nesta mensagem a melhor configuração para estabelecer a segurança da conexão.

3. *ServerCertificate* - Nesta mensagem o servidor envia o seu *certificado*, no formato padrão X.509.

Figura 10 – Representação do Protocolo Handshake.



Fonte: Autoria própria

4. *ServerKeyExchange* - Esta mensagem é enviada para o cliente e contém informações necessárias para que este seja capaz de calcular e trocar um segredo, conhecido também como chave pré-*master*.

5. *ServerHelloDone* - Mensagem enviada pelo servidor, que informa ao cliente que o processo de negociação foi finalizado.

6. *ClientKeyExchange* - Mensagem enviada do cliente para o servidor servidor, para que este último também tenha condições de gerar a chave secreta à ser utilizada pelo protocolo de registro.

7. *ChangeCipherSpec* - Enviada do cliente para o servidor à fim de informar que o conjunto de cifras à ser utilizado deve ser atualizado.

8. *ClientFinished* - O cliente envia esta mensagem ao servidor para informar que finalizou sua parte no *handshake*.

9. *ChangeCipherSpec* - Mensagem enviada do servidor para o cliente informando que o estado pendente, do conjunto de cifras, foi atualizado para o estado atual.

10. *ServerFinished* - Enviada ao cliente ara finalizar o *handshake* (CENTER, 2016).

Na mensagem *ClientHello* o cliente envia uma requisição de conexão para o servidor, na qual possui seus parâmetros de configurações para que o servidor possa negociar o

conjunto de cifras, versão do protocolo, um número aleatório, um ID de sessão e algoritmos de compressão. O número aleatório possui 32 bytes, sendo 28 aleatórios e 4 utilizados para carregar informações do relógio do cliente (em segundos), é utilizado para evitar ataques de repetição. O ID é um número de 32 bytes gerado aleatoriamente e é utilizado para identificar cada sessão, caso este campo seja nulo, significa que o cliente deseja iniciar uma nova sessão, do contrário, é utilizado com o valor que indica exatamente o ID da sessão que o cliente deseja retomar e, por fim, o algoritmo de compressão é nulo, por padrão.

Ao receber a mensagem *ClientHello* o servidor negocia, de acordo com os parâmetros enviados pelo cliente, a melhor configuração para estabelecer a segurança da conexão. Decide a melhor versão do protocolo, conjunto de cifras, algoritmo de compressão e envia também um número aleatório, para evitar o ataque de repetição. Retorna um número aleatório como ID da sessão, caso este campo estiver vazio e envia informações adicionais para informar se este oferece suporte para a renegociação de sessão (alteração do conjunto de cifras). Estas informações são enviadas para o cliente na mensagem *ServerHello*.

Em seguida, o servidor envia o seu certificado (em *ServerCertificate*), no formato padrão X.509, para que o cliente possa autenticá-lo. O envio do certificado não é obrigatório, pois existem outros métodos de autenticação que não requer certificados, como por exemplo, chaves *PGP* (*Pretty Good Privacy*) (Network Associates, 1999), mas este último não será discutido pois todos os sítios analisados para o trabalho utilizam certificados X.509. Caso o servidor exija a autenticação do cliente, este é o momento em que a solicitação é enviada.

O cliente, ao receber o certificado digital enviado pelo servidor, verifica se as informações anexadas são confiáveis, ou seja, se o certificado é válido, confiável e se ele está, de fato, relacionado ao seu proprietário. A verificação ocorre da seguinte maneira: o cliente tem informações sobre a chave pública do assinante do certificado (se ela for de uma AC reconhecida) e a utiliza para decriptografar a assinatura (o *hash*) que está anexado ao certificado.

No certificado também há informações sobre o algoritmo de *hash* utilizado para assiná-lo, de maneira que o cliente aplica o mesmo algoritmo no certificado e compara o *hash* gerado com o *hash* que foi decriptografado. Caso os *hashes* sejam idênticos, então este certificado está verdadeiramente relacionado ao seu proprietário. É verificado também o período de validade e informações adicionais, como por exemplo, se o certificado já foi revogado.

Após enviar seu certificado digital, o servidor envia a mensagem *ServerKeyExchange* para o cliente, esta contém informações necessárias para que o cliente seja capaz de calcular e trocar um segredo, conhecido também como chave pré-*master*, que tem por objetivo realizar um teste para verificar se a conexão poderá ser estabelecida de maneira confiável, uma vez que a chave secreta não é calculada diretamente. Esta mensagem é enviada apenas quando se utilizam os algoritmos para a troca de chaves como o *Diffie-Hellman Anônimo*, por exemplo. E, em seguida, envia a mensagem *ServerHelloDone*, para informar ao cliente

que o processo de negociação foi finalizado e que está aguardando por retorno.

Para enviar a mensagem *ClientKeyExchange* o cliente depende do conjunto de cifras negociados anteriormente, para contruibuir na criação de um segredo para a troca de chaves. Após validar o certificado e receber as informações sobre o segredo(*pre-master* que deve ser gerado, o cliente envia informações para que o servidor também tenha condições de gerar a chave secreta à ser utilizada pelo protocolo de registro. Esta mensagem é criptografada com a chave pública presente no certificado e enviada ao servidor.

Caso o servidor tenha solicitado ao cliente o envio do seu certificado, este deve ser enviado, mas se o cliente não enviar, então é enviada ao servidor uma mensagem de *no_certificate*, tratada pelo protocolo de alertas discutido na Sessão 3.1.2.

Como ambas as partes agora conhecem as informações necessárias para calcular a chave simétrica, assim o fazem.

Depois de gerar a chave simétrica à ser utilizada pelo protocolo de registro, o cliente envia a mensagem *ChangeCipherSpec* ao servidor para informá-lo de que o conjunto de cifras à ser utilizado deve ser atualizado. De maneira que o estado pendente se torna o estado atual.

Quando o cliente envia a mensagem *ClientKeyExchange* e recebe um retorno do servidor, ocorre a autenticação do servidor, pois o cliente tem a prova de que realmente está se comunicando com o servidor quem diz ser porque só ele seria capaz de decifrar, com sua chave privada, a mensagem enviada pelo cliente uma vez que ele possui o par da chave pública presente no certificado.

Por fim, o cliente envia a *ClientFinished* ao servidor para informar que finalizou sua parte no *handshake*, então servidor envia a *ChangeCipherSpec* como resposta, informando que o estado pendente foi atualizado para o estado atual, seguido da mensagem *ServerFinished* que é enviada ao cliente à fim de comunicar que houve a mudança no estado do conjunto de cifras e que este está pronto para ser utilizado, e então finaliza o protocolo *handshake* (CENTER, 2016).

Depois da sessão estabelecida e de negociado os parâmetros do conjunto de cifras, o protocolo de registro já tem condições de utilizá-lo para cifrar as mensagens que serão enviadas.

3.1.2 Protocolo de Registro

O protocolo de registro TLS é utilizado para estabelecer a comunicação e utiliza os parâmetros de configuração estabelecidos no *handshake* para executar o processo de encriptação das mensagens, à fim de padronizar e assegurar a *integridade* e *confidencialidade* das informações à serem enviadas para protocolos da camada superior que opera com o

TLS, como por exemplo, para o HTTP (STALLINGS, 2008).

As mensagens à serem enviadas, tanto pelo emissor, quanto pelo receptor, são padronizadas em blocos de tamanho fixo que recebem um MAC, para que ao receber a mensagem o receptor consiga verificar a autenticidade e integridade desta. Este MAC é criptografado com a chave secreta que foi gerada durante o *handshake* e, recebe um cabeçalho TLS, que contém informações das versões do protocolo que estão sendo utilizadas, bem como informações do tamanho da mensagem. O processo apresentado na Figura 11 é realizado para cada mensagem da seguinte maneira:

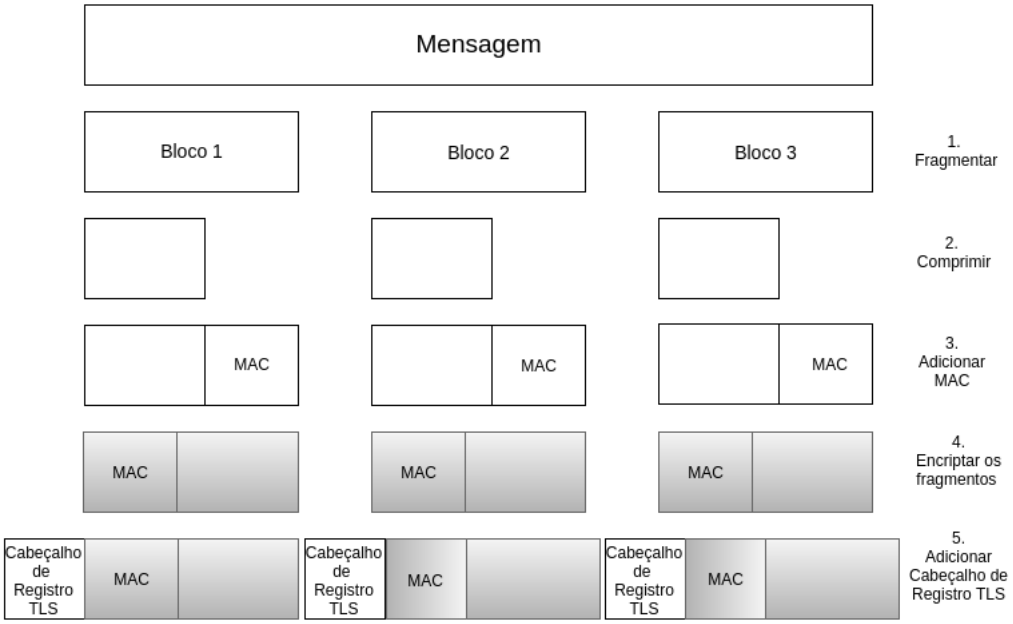
1. A mensagem é dividida em blocos de tamanhos iguais, caso falte mensagem para completar o tamanho do bloco, este deve ser preenchido utilizando bits de valor 0;
2. Os fragmentos da mensagem são compactados ou não;
3. É adicionado um código de autenticação de mensagem (MAC) aos fragmentos da mensagem;
4. Os fragmentos da mensagem, com o cabeçalho são criptografados utilizando criptografia simétrica, utilizando a chave que foi calculada no *handshake*;
5. Adicionar um cabeçalho de registro TLS;

Ao receber a mensagem o receptor utiliza mesma chave secreta (usada para cifrar) para decifrar a mensagem e o MAC, neste momento ocorre a autenticação da mensagem, bem como a sua decriptografia. Em seguida, é executado na mensagem o mesmo algoritmo de *hash* utilizado pelo emissor, para que o receptor possa comparar o *hash* gerado, com o *hash* recebido. Caso sejam idênticos, a mensagem recebida é íntegra.

Após o processo descrito é possível observar que o servidor é autenticado durante o protocolo *handhsake* e, que o protocolo de registro garante que as mensagens sejam enviadas em sigilo, sejam íntegras e autenticadas. Para esta demonstração, a etapa de compressão (opcional) foi ignorada.

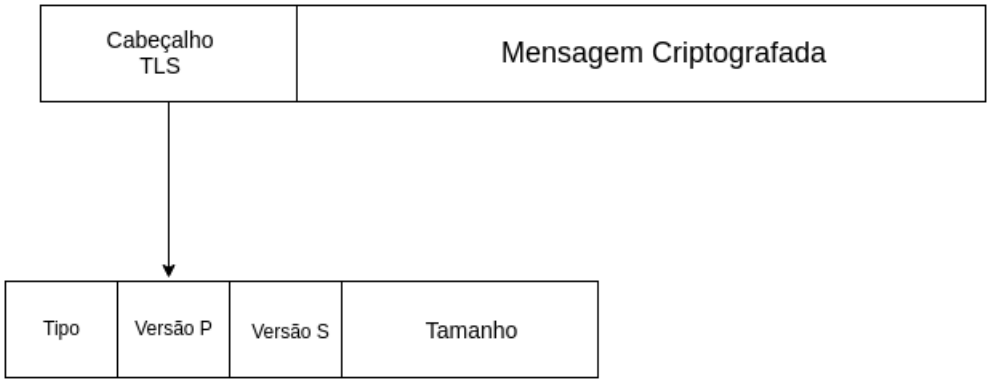
O tamanho dos blocos da mensagem deve ser, no máximo, de 2^{14} bytes (16384 bytes). A compactação deve ocorrer sem perdas, para não alterar o significado da mensagem, e não exceder o tamanho do conteúdo em mais que 1024 bytes. A chave secreta gerada pelo protocolo *handshake* é utilizada no algoritmo que calcula o MAC. Todo cabeçalho de registro TLS contém campos que informam o *tipo de conteúdo*, que pode ser *handshake*, *change_cipher_spec* ou *alert_protocol*, a *versão principal* e *secundária*, que contém a informação das versões do TLS que são utilizadas, e o *tamanho do fragmento*, que compactado (ou não) e criptografado não deve exceder à 2^{14} + 2048 bytes (STALLINGS, 2008). A representação do cabeçalho de registro TLS é visto na Figura 12.

Figura 11 – Representação da estrutura do Protocolo de Registro TSL.



Fonte: Autoria própria

Figura 12 – Representação da estrutura do cabeçalho TLS adicionado à um bloco de mensagem criptografado.



Fonte: Autoria propria

- **Protocolo de Mudança de Cifra**

  O protocolo de mudança de cifra, ou *change_cipher_spec*, consiste em apenas uma mensagem, com um único byte, contendo o valor 1, e indica que um estado pendente seja copiado para o estado atual, fazendo com que o conjunto de cifras utilizado nessa conexão seja atualizado. O conjunto de cifras é a configuração que garante a segurança da troca das mensagens, nele contém o algoritmo de criptografia simétrica, o algoritmo de *hash*, o tamanho do *hash* e da chave à serem utilizados em uma conexão (STALLINGS, 2008).

- **Protocolo de Alerta**

  O protocolo de alerta, ou *alert_protocol*, o é responsável por enviar alertas às partes

relacionadas na comunicação que está utilizando o TLS. Ele tem apenas dois campos, de um byte cada um, para indicar o tipo de alerta e a sua descrição. Podem ser considerados como alertas fatais, setando no tipo de alerta um byte com o valor 2 ou, como aviso, setando o byte de alerta com o valor 1. Os alertas fatais fazem com que a conexão seja encerrada imediatamente, pois pode comprometer a segurança sessão, dessa forma, a sessão pode continuar com as conexões que já tinha estabelecido antes de receber o alerta fatal, mas não pode estabelecer mais nenhuma.
São exemplos de alertas (*warnings*):

- ***no_certificate***: retorna a informação de que não foi encontrado nenhum certificado apropriado, caso o certificado emitido não tenha a assinatura de uma CA reconhecida, ou se o período de validade expirou;
- ***close_notify***: é enviado ao servidor, informando que o cliente irá encerrar com a conexão, ou seja, não irá mais enviar mensagens e ambas as partes devem enviar este alerta para finalizar a conexão;

São exemplos de alertas fatais:

- ***handshake_failure***: ocorre quando, por exemplo, um servidor exige a autenticação do cliente, mas o cliente não tem nenhum certificado para retornar, então esta mensagem é apresentada ao servidor pois houve falha na negociação dos parâmetros de segurança que foram disponibilizados pelo servidor;
- ***decompression_failure***: é emitido quando a mensagem a ser descompactada ultrapassa o valor do tamanho permitido, ou por receber algum tipo de entrada que não seja adequada (STALLINGS, 2008) (RISTIĆ, 1999);

Um servidor TLS se não for configurado da maneira correta é considerado inseguro e pode estar sujeito à diversos tipos de ataques (RISTIĆ, 2014b). No próximo Capítulo é apresentado alguns destes ataques e os parâmetros de configuração que devemos verificar para usar o TLS seguro.

CAPÍTULO 4

Segurança do TLS

Para usar o TLS seguro é necessário ter atenção na escolha dos parâmetros de configuração. A configuração errada em vez de garantir a segurança de um servidor TLS, pode ser uma verdadeira armadilha e, configurá-lo da maneira correta não significa que o servidor estará totalmente seguro, mas que pode dificultar muito o trabalho de atacantes. Vários aspectos são relevantes para identificar a segurança de um servidor TLS, dentre eles: o certificado digital, versões do protocolo, tamanho das chaves, algoritmo de *hash* e conjunto de cifras (RISTIĆ, 2014b).

Os certificados carregam informações importantes como a chave pública do proprietário e atributos do proprietário, o seu período de validade e a assinatura digital da AC que o emitiu (NAKAMURA; GEUS, 2007). Um certificado é considerado *inseguro* se ele é auto assinado ou se foi assinado por uma outra entidade que não seja uma AC reconhecida pelo *browser*, se o algoritmo de *hash* utilizado para a assinatura for o MD2, MD5 ou SHA1, se o tamanho da chave for menor que 2048-bits, ou se o nome de domínio é incompatível com o certificado, ou seja, um nome está configurado para resolver aquele dado IP, mas este nome não está presente no certificado. E, por fim, é considerado *inválido* o certificado que tiver o período de validade excedido ou se já foi revogado (RISTIĆ, 2014b).

As chaves estão diretamente ligadas com a segurança da comunicação, são as responsáveis por cifrar/decifrar as mensagens enviadas pelos clientes e a dificuldade em descobri-la está diretamente ligada ao seu tamanho. Quanto maior a chave, mais difícil será quebrá-la. Em relação às chaves privadas, estas devem ser mantida em sigilo absoluto, pois caso alguém tenha seu conhecimento poderá usá-la e se passar pelo proprietário real sem que o cliente saiba, comprometendo assim toda a segurança da comunicação. Chaves assimétricas com tamanho menor que 2048-bits são consideradas inseguras portanto, recomenda-se o uso de chaves com tamanho de 2048-bits ou 256-ECDSA (RISTIĆ, 2014b).

O algoritmo de *hash* é utilizado para assinar o certificado digital e embora o SHA1 seja o mais utilizado, é recomendado que se utilize o SHA256. A diferença entre o *hash* SHA1 e o SHA256 está na probabilidade de ocorrer colisões, ou seja, obter dois *hashes* idênticos

para dois textos diferentes. Ambas as funções utilizam, como entrada, blocos de 512 bits, mas possuem saídas com tamanho diferentes, 160-bits e 256-bits, respectivamente. A probabilidade de ocorrer colisões quando se usa o SHA1 é muito menor se comparada ao SHA256 (MORTON; SMITH, 2014) (INTERNET, 2016).

Dessa forma, para que a comunicação seja estabelecida de maneira segura, é necessário configurar o protocolo TLS atendendo aos requisitos fundamentais para garantir a segurança:

- *Tamanho das Chaves*: É necessário utilizar chaves de tamanho adequado. É certo que quanto maior a chave utilizada, mais difícil será descobri-la, mas chaves grandes demais significa um custo elevado de processamento, tornando o processo de cifragem-decifragem mais lento. Atualmente, recomenda-se usar chaves de tamanho 2048-bits para o algoritmo RSA ou, caso seja necessário utilizar chave de tamanho maior, usar o ECDSA, de 256-bits, apesar de nem todos os clientes suportarem este último. Chaves com menos de 1024-bits devem ser alteradas para 2048-bits (RISTIĆ, 2014b).

- *Função de Hash* segura para os Certificados: Os certificados são assinados por meio de *hash*, dessa maneira, para que o certificado seja seguro é necessário utilizar um algoritmo de *hash* seguro. O mais utilizado é o SHA1, cujo uso não é recomendado, pois a partir do final do ano de 2016 dará lugar ao seu sucessor, SHA2 (RISTIĆ, 2014b).

- *Versão do Protocolo*: Há cinco versões do protocolo: SSLv2, SSLv3, TSLv1.0, TLSv1.1 e TLSv1.2. O protocolo SSLv2 é um protocolo obsoleto e o SSLv3 é inseguro por ser vulnerável à ataques como o *POODLE*, por exemplo. O SSLv3 não deve ser implementado junto com os seus sucessores, pois pode comprometer a segurança da comunicação, por meio do *downgrade*, que consiste no bloqueio intencional (feito por um cliente) dos protocolos sucessores: TLS1.0, 1.1 e 1.2, obrigando o servidor a se conectar com a versão SSL3 (MOELLER; LANGLEY, 2014) (RISTIĆ, 2014b). Logo, as versões SSL são consideradas inseguras e o suporte à estas versões deve ser evitado. Recomenda-se utilizar a versão mais recente (TLSv1.2), pois é a versão mais atual e conta com recursos que estão ausentes nas versões anteriores. Apesar de ser a versão mais atual, alguns clientes não possuem suporte ao TLSv1.2, sendo necessário deixar habilitado as versões TLSv1.0 e TLSv1.1, que também são consideradas seguras por não terem nenhuma falha de segurança conhecida até o momento (RISTIĆ, 2014b).

- *Certificado*: Deve ser assinado por uma AC reconhecida e confiável, estar dentro do prazo de validade, não ter sido revogado. Neste devem ser especificados todos os nomes que resolvem o IP do servidor TLS, para que não sejam emitidos, aos usuários, mensagens de alertas relacionados à falta de confiança no certificado (RISTIĆ, 2014b).

- É importante *desativar a renegociação de parâmetros* que podem ser solicitados pelo cliente, de maneira que só o servidor possa fazer a solicitalção. Deixar que o cliente inicie a renegociação poderá fazer com que o servidor fique suscetível à ataques de negação de serviço (*DOS*) (RISTIĆ, 2014b).

- Utilizar *conjuntos de cifras seguros*, pois são estes que definem o quão segura uma comunicação deve ser. Para tanto, deve ser usada criptografia de 128-bits ou mais, evitar o algoritmo *Diffie-Hellman* anônimo para a troca de chaves, evitar conjuntos de cifras nulos (sem nenhuma configuração), evitar conjuntos de cifras fracos: que utilizam 40-bits e 56-bits e, não utilizar a criptografia RC4, por também ser considerado fraco e estar sujeito à ataques por milhões de requisições (RISTIĆ, 2014b).

- *Desativar a compressão de dados*, pois é vulnerável ao CRIME ataque, discutido na Sessão 4.1, no qual o atacante utiliza deste método para descobrir informações sigilosas (RISTIĆ, 2014b).

4.1 Ataques

Os ataques podem ser classificados em *passivos* e *ativos*. Em ataques *passivos* o atacante tem o acesso às mensagens que deveriam ser sigilosas, mas não alteram seu conteúdo, e em ataques *ativos* além do acesso inautorizado, o atacante altera seu conteúdo e se passa por um cliente à fim de obter alguma vantagem. Para evitar estes ataques o TLS propõe a garantia da comunicação segura ponto-a-ponto e quando não se tem esta implementação ou, quando implementado, permite o acesso sem a autenticação (HTTP), o servidor *web* fica mais vulnerável ao ataque *MITM* (*Man-In-The-Middle*), pois sem a autenticação, o cliente não pode ter a certeza absoluta de que está se conectando ao servidor verdadeiro (DIERKS T.; ALLEN, 2009). Os ataques ao TLS mais conhecidos são:

- *MITM attack*: Consistem em um atacante se infiltrar em uma comunicação ponto-a-ponto entre dois agentes diferentes, interceptar as informações trocadas e se passar por uma das partes. Um atacante pode conseguir o acesso à comunicação da seguinte maneira: *A* deseja estabelecer uma comunicação com *B* então, *A* envia informações necessárias à *B*, para que este seja capaz de calcular a chave secreta que será utilizada na sessão. Mas há um *C* que deseja interceptar as informações entre *A* e *B*, dessa forma, *C* intercepta as informações enviadas por *A*, calcula a chave secreta e estabelece uma sessão com *A*, então, *C* faz o mesmo procedimento com *B*. Logo, *A* envia mensagens para *C*, quando na verdade deveria enviar mensagens para *B*. *C*, por sua vez, se passa *A* e também pode se comunicar com *B*, como se este terceiro não existisse (TANENBAUM, 2002).

- *CRIME attack*: do inglês *Compression Ratio Info-leak Made Easy*, é um ataque de força bruta ao TLS, baseado na compressão de dados, que tem por objetivo espiar as sessões, enviando várias requisições HTTP ao cliente, à fim de descobrir informações sobre tokens de sessões ou outras informações sigilosas. Os ataques podem ocorrer quando o algoritmo de compressão *gzip* ou *DEFLATE* for usado. É por isso que, apesar de tornar o carregamento da página mais rápido, por padrão, o algoritmo de compressão do TLS é nulo, não é utilizado (SARKAR; FITZGERALD, 2013).

- *POODLE attack*: em outubro de 2014 foi publicada uma nota sobre a vulnerabilidade do protocolo SSLv3 ao ataque *POODLE*, do inglês, *Padding Oracle On Downgraded Legacy Encryption*, que consiste em um atacante (*man-in-the-middle*) observar o tráfego no canal de comunicação entre um cliente e servidor e, no momento do handshake bloquear, de maneira intencional, a conexão por meio dos protocolos TLS, obrigando o servidor à estabelecer a conexão utilizando a versão SSLv3 (quando um servidor tenta a conexão utilizando um protocolo mais recente e não obtém sucesso, este tenta se conectar com a versão mais antiga, no caso, o SSL3.0), dessa forma, o servidor terá que utilizar conjuntos de cifras fracos, como a criptografia RC4, ou cifras de bloco CBC, que facilitam a exposição dos dados encriptados. Portanto, o ataque *POODLE* é um ataque *MITM*, seguido de um *downgrade*, os quais permitem o acesso à informações sensíveis. Este ataque ocorre somente na versão SSLv3 (CABALLERO *et al.*, 2016) (MOLLER *et al.*, 2014).

CAPÍTULO 5

Avaliação do Uso do TLS na Rede da UFG

Neste capítulo são discutidos os parâmetros utilizados para verificar a segurança do TLS nas estações servidoras da UFG, bem como a metodologia adotada para auxiliar na análise e obter os resultados quanto aos servidores que são seguros ou não e o porquê.

5.1 Metodologia

Para realizar a avaliação da segurança do TLS foram escolhidos quatro parâmetros à serem verificados em 66 estações servidoras, estas que foram resultados de uma varredura na rede da Universidade Federal de Goiás (UFG), por meio da filtragem de servidores que aceitam conexão na porta 443.

De acordo com as melhores práticas do TLS, discutidas no capítulo 04, os quatro parâmetros de segurança utilizados para a avaliação das estações servidoras foram escolhidos com base na autenticação do servidor e na segurança fornecida pela versão do protocolo, são eles: tamanho da chave privada, algoritmo de *hash*, validade do certificado e versões do protocolo TLS. A autenticação do servidor e a versão do protocolo utilizada é o passo inicial para que uma sessão seja estabelecida de maneira segura, pois, por exemplo, não adianta usar um conjunto de cifras considerado forte se não houver cuidado quanto à escolha do tamanho da chave privada ou, configurar todos os parâmetros de maneira segura, mas oferecer suporte à versão do protocolo SSLv3, que é vulnerável ao ataque *POODLE*. As configurações para o uso seguro do TLS são:

- Ter *Chave privada com tamanho* de 2048-bits.
- O *Algoritmo de hash* utilizado deve ser o SHA256.
- Utilizar *versões do protocolo TLS* e não oferecer suporte às versões SSL.

- O *Certificado do servidor* deve ser válido e confiável.

As ferramentas utilizadas para as avaliações foram:

- *SSLLABS-SCAN*, versão 1.3.0. Implementada por Ivan Ristić, para a API SSL Labs, foi lançada em dezembro de 2009 e disponibilizada pela *Qualys*[1]. É uma ferramenta *open source*, executada via linha de comando, que analisa um servidor TLS, público, por meio da simulação do *handshake* para se conectar com um servidor TLS e, como resultado, gera um arquivo com extensão *.json* que contém as informações de configurações do TLS que poderiam ser utilizadas para estabelecer uma sessão com um cliente (*browser*), tais como: IP que resolve DNS, o nome do domínio, nomes alternativos que podem ser utilizados, certificado no padrão X.509, algoritmo de *hash*, tamanho das chaves, versões do protocolo TLS, conjunto de cifras, dentre outras.

- *SSL-LUFG*, ferramenta implementada em Java e desenvolvida durante o trabalho, é utilizada para analisar os arquivos, *.json*, gerados pela *ssllabs-scan* à fim de extrair informações quanto à chave privada, algoritmo de *hash*, validade do certificado (se o prazo de validade foi excedido, se é auto assinado) e versões do protocolo que o servidor utiliza. Após analisar os parâmetros de segurança, é apresentado um relatório com as inferências quanto à segurança dos servidores TLS. Os arquivos *.json* são lidos e analisados pela ferramenta *ssl-lufg* conforme segue:

1. Tamanho da chave RSA (o parâmetro *keySize* é utilizado para verificar se o tamanho da chave é de 1024-bits ou 2048-bits):
   "endpoints.0.details.chain.certs.0.keySize": 1024
   ou
   "endpoints.0.details.chain.certs.0.keySize": 2048

2. Algoritmo de *hash* utilizado (o parâmetro *sigAlg* é utilizado para verificar se o algoritmo de *hash* é o SHA1 ou SHA256):
   "endpoints.0.details.cert.sigAlg": "SHA1withRSA"
   ou
   "endpoints.0.details.cert.sigAlg": "SHA256withRSA"

3. Protocolo(s) utilizado: SSL2, SSL3, TLS1.0, TLS1.1 ou TLS1.2 (os parâmetros *name* e *version* são utilizados para verificar qual versão do protocolo é usada:
   "endpoints.0.details.protocols.0.name": "TLS"
   "endpoints.0.details.protocols.0.version": "1.0"
   ou
   "endpoints.0.details.protocols.0.name": "TLS"

[1] Link disponível para download: https://github.com/ssllabs/ssllabs-scan/releases

"endpoints.0.details.protocols.0.version": "1.1"
ou
"endpoints.0.details.protocols.0.name": "TLS"
"endpoints.0.details.protocols.0.version": "1.2"
ou
"endpoints.0.details.protocols.0.name": "SSL"
"endpoints.0.details.protocols.0.version": "3"

4. Nome do domínio, ou *commonNames* (CN):
   Exemplo utilizando o servidor TLS *www.inf.ufg.br* (CN):
   "endpoints.0.details.cert.commonNames.0": "www.inf.ufg.br"

5. Período de Validade do Certificado:
   O período de validade é dado em segundos desde o ano 1970. Foi necessário realizar o cálculo de conversão para identificar a data em dias / anos / horas. São analisados os parâmetros *notAfter* e *notBefore* para verificar o período de validade: não é válido antes da data especificada em *notBefore* e nem depois da data especificada em *notAfter*.
   Exemplo:
   "endpoints.0.details.cert.notAfter": 1515414567000
   "endpoints.0.details.cert.notBefore": 1420806567000

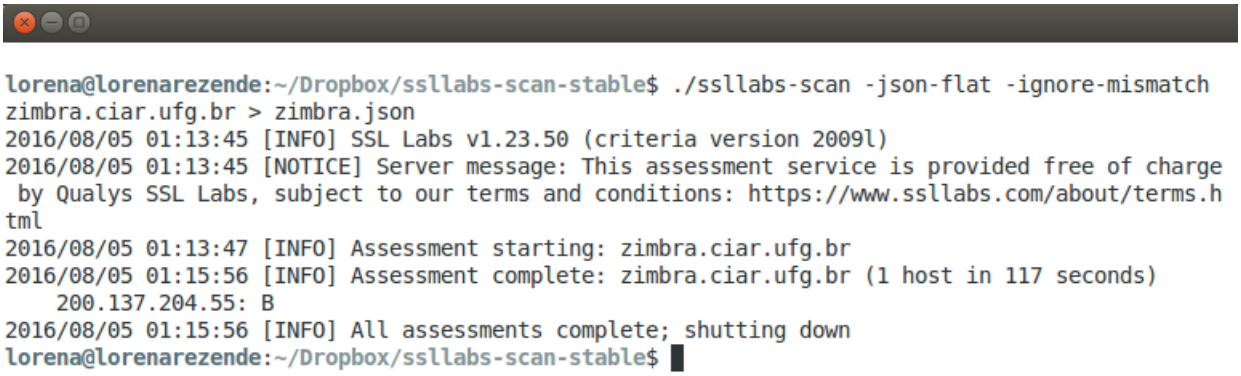
6. Assinatura do Certificado (se é auto assinado ou não):
   Necessário verificar se em *Issuer* os parâmetros são idênticos aos que estão em *Subject*, se SIM, o certificado é auto assinado. Exemplo de um certificado auto assinado:
   SUBJECT: "endpoints.0.details.cert.subject":
   "1.2.840.113549.1.9.1=16127375706f72746540696e662e7566672e6272,
   CN=www.inf.ufg.br,OU=Instituto de Informatica,O=Universidade Federal de Goias,
   L=Goiania,ST=Goias,C=BR"
   ISSUER: "endpoints.0.details.chain.certs.0.issuerSubject":
   "1.2.840.113549.1.9.1=16127375706f72746540696e662e7566672e6272,
   CN=www.inf.ufg.br,OU=Instituto de Informatica,O=Universidade Federal de Goias,
   L=Goiania,ST=Goias,C=BR"

Os campos: *"endpoints.0.details.chain.certs.0.keySize"*, *"endpoints.0.details.cert.sigAlg"*, *"endpoints.0.details.protocols.0.name"*, *"endpoints.0.details.protocols.0.version"*, *"endpoints.0.details.cert.commonNames.0"*, *"endpoints.0.details.cert.notAfter"*, *"endpoints.0.details.cert.notBefore"*, *SUBJECT: "endpoints.0.details.cert.subject"*, *ISSUER: "endpoints.0.details.chain.certs.0.issuerSubject"*, *CN*, *OU*, *O*, *L*, *ST* e *C* são informações contidas nos arquivos *.json* e que são utilizadas para as verificações.

5.1.1 Utilização da ferramenta *ssllabs-scan* e da ferramenta desenvolvida

Para gerar o arquivo *.json* das estações servidoras na rede da UFG, foi utilizado o comando *./ssllabs-scan* com os parâmetros *-json-flat -ignore-mismatch* para que a simulação do *handshake* continuasse, mesmo se o certificado fosse incompatível com o nome de domínio que foi usado. A Figura 13 representa a utilização da ferramenta *ssllabs-scan* e o servidor TLS exemplo é o *zimbra.ciar.ufg.br*

Figura 13 – Simulação do protocolo *handshake* utilizando a ferramenta *ssllabs-scan*, tomando como exemplo a estação servidora *zimbra.ciar.ufg.br*.



Fonte: Autoria própria

O tempo de execução da ferramenta *ssllabs-scan* para verificar o servidor TLS *zimbra.ciar.ufg.br* foi de 117 segundos. O arquivo *.json* gerado para este exemplo pode ser visto no Anexo *A.1*.

Após a execução da ferramenta *ssllabs-scan* foi utilizada a *ssl-lufg*, para realizar a varredura dos arquivos *.json* e informar se o servidor TLS é seguro ou não e o porquê. Na Figura 14 tem-se a apresentação do resultado gerado pela ferramenta *ssl-lufg*:

Figura 14 – Inferência sobre o arquivo *.json* gerado pela ferramenta *ssllabs-scan*, tomando como exemplo a estação servidora *zimbra.ciar.ufg.br*.

```
****zimbra.ciar.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ciar.ufg.br"
Proprietário(subject): "CN=*.ciar.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Mar 01 11:21:05 BRT 2016 - Sat Mar 02 11:21:05 BRT 2019Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 3 requisitos de segurança TLS!
O servidor é inseguro porque o certificado é auto assinado
```

Fonte: Autoria própria

5.2 Resultados

De todos os 66 servidores analisados foi constatado que para 11 deles ocorreu *timeout* na tentativa de estabelecer a conexão TLS, de maneira que não puderam ter suas configurações verificadas, são eles:

1. *h20013720480.ufg.br* (200.137.204.80)
2. *h200137204119.ufg.br* (200.137.204.119)
3. *h20013720571.ufg.br* (200.137.205.71)
4. *h20013721753.ufg.br* (200.137.217.53)
5. *h200137217204.ufg.br* (200.137.217.204)
6. *h200137218134.ufg.br* (200.137.218.134)
7. *h2001372199.ufg.br* (200.137.219.9)
8. *h200137219112.ufg.br* (200.137.219.112)
9. *www.gestaodenegocios.eeec.ufg.br*
10. *h200137221167.ufg.br* (200.137.221.167)
11. *h200137221188.ufg.br* (200.137.221.188)

Os 55 servidores que aceitaram a conexão TLS, foram classificados em seguros ou inseguros, de acordo com os quatro parâmetros de segurança utilizados para a avaliação: tamanho da chave, algoritmo de *hash*, versão do protocolo e certificado.

5.2.1 Protocolos inseguros e suas especificações

São considerados como *inseguros* os servidores TLS que tem chaves menores que 2048-bits, algoritmo de *hash* que não seja o SHA256, se oferece suporte à versões do protocolo SSL e se o certificado for inválido ou inseguro: auto assinado (salvo casos em que o certificado pertence à uma autoridade certificadora reconhecida pelo usuário (*browser*)), se é assinado por uma AC desconhecida, se já houve revogação, se o certificado é incompatível (quando o nome do domínio é diferente do nome digitado no *browser*), ou se o período de validade foi excedido.

De acordo com estas informações, os servidores inseguros foram subdivididos de acordo com o seu conjunto de configurações:

- Tamanho da chave privada: 1024-bits
  Algoritmo de *hash*: SHA1
  Certificado: auto assinado e dentro do prazo de validade
  Versão do protocolo: TLSv1.0
  No Quadro 1 estão os servidores TLS inseguros com as respectivas configurações.

Quadro 1 – Servidores com chave privada de tamanho 1024-bits, algoritmo de *hash* SHA1, com certificado auto assinado e dentro do prazo de validade. Versão do protocolo: TLSv1.0

	Servidor
1	artemis.inf.ufg.br
2	Mercurio-2.ciar.ufg.br
3	horde5.inf.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 1024-bits
  Algoritmo de *hash*: SHA1
  Certificado: auto assinado e prazo de validade excedido (inválido)
  Versão do protocolo: TLSv1.0
  No Quadro 2 está o servidore TLS inseguro com as respectivas configurações.

Quadro 2 – Servidor com chave privada de tamanho 1024-bits, algoritmo de *hash* SHA1, com certificado auto assinado e prazo de validade excedido. Versão do protocolo: TLSv1.0

	Servidor
1	hades.inf.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 1024-bits
  Algoritmo de *hash*: SHA1
  Certificado: auto assinado e dentro do prazo de validade
  Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0
  No Quadro 3 estão os servidores TLS inseguros com as respectivas configurações.

Quadro 3 – Servidores com chave privada de tamanho 1024-bits, algoritmo de *hash* SHA1, com certificado auto assinado e dentro do prazo de validade. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0

	Servidor
1	ead.inf.ufg.br
2	h20013720429.ufg.br
3	dionisio.inf.ufg.br
4	projetos.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 2048-bits
  Algoritmo de *hash*: SHA1
  Certificado: auto assinado e dentro do prazo de validade
  Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0
  No Quadro 4 estão os servidores TLS inseguros com as respectivas configurações.

Quadro 4 – Servidores com chave privada de tamanho 2048-bits, algoritmo de *hash* SHA1, com certificado auto assinado e dentro do prazo de validade. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0

	Servidor
1	mail.labora.inf.ufg.br
2	mail.cpa.evz.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 2048-bits
  Algoritmo de *hash*: SHA1
  Certificado: auto assinado e prazo de validade excedido (inválido)
  Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0
  No Quadro 5 está o servidor TLS inseguro com as respectivas configurações.

Quadro 5 – Servidor com chave privada de tamanho 2048-bits, algoritmo de *hash* SHA1, com certificado auto assinado e prazo de validade excedido. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0

	Servidor
1	sgbd.inf.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 2048-bits
  Algoritmo de *hash*: SHA1
  Certificado: não é auto assinado e está dentro do prazo de validade
  Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0
  No Quadro 6 estão os servidores TLS inseguros com as respectivas configurações.

Quadro 6 – Servidores com chave privada de tamanho 2048-bits, algoritmo de *hash* SHA1, com certificado que não é auto assinado e está dentro do prazo de validade. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0

	Servidor
1	cia2.inf.ufg.br
2	h200137217163.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 2048-bits
  Algoritmo de *hash*: SHA1

Certificado: não é auto assinado e tem prazo de validade excedido (inválido)

Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0

No Quadro 7 estão os servidores TLS inseguros com as respectivas configurações.

Quadro 7 – Servidor com chave privada de tamanho 2048-bits, algoritmo de *hash* SHA1, com certificado que não é auto assinado e tem prazo de validade foi excedido. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0

	Servidor
1	h200137221168.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 2048-bits

  Algoritmo de *hash*: SHA256

  Certificado: auto assinado e tem prazo de validade excedido (inválido)

  Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0 e SSLv3

  No Quadro 8 está o servidor TLS inseguro com as respectivas configurações.

Quadro 8 – Servidor com chave privada de tamanho 2048-bits, algoritmo de *hash* SHA256, com certificado auto assinado e prazo de validade excedido. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0 e SSLv3

	Servidor
1	h20013721675.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 2048-bits

  Algoritmo de *hash*: SHA256

  Certificado: não é auto assinado e tem prazo de validade excedido (inválido)

  Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0 e SSLv3

  No Quadro 9 está o servidor TLS inseguro com as respectivas configurações.

Quadro 9 – Servidor com chave privada de tamanho 2048-bits, algoritmo de *hash* SHA256, com certificado que não é auto assinado e tem prazo de validade excedido. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0 e SSLv3

	Servidor
1	h200137217126.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 2048-bits

  Algoritmo de *hash*: SHA1

  Certificado: auto assinado e tem prazo de validade excedido (inválido)

  Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0 e SSLv3

  No Quadro 10 está o servidor TLS inseguro com as respectivas configurações.

Quadro 10 – Servidor com chave privada de tamanho 2048-bits, algoritmo de *hash* SHA1, com certificado auto assinado e prazo de validade excedido. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0 e SSLv3

	Servidor
1	h20013721677.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 1024-bits
  Algoritmo de *hash*: SHA1
  Certificado: auto assinado e prazo de validade excedido (inválido)
  Versão do protocolo: TLSv1.2
  No Quadro 11 está o servidor TLS inseguro com as respectivas configurações.

Quadro 11 – Servidor com chave privada de tamanho 1024-bits, algoritmo de *hash* SHA1, com certificado auto assinado e prazo de validade foi excedido. Versões do protocolo: TLSv1.2

	Servidor
1	ns1.sectec.go.gov.br

Fonte: Autoria propria

- Tamanho da chave privada: 1024-bits
  Algoritmo de *hash*: SHA1
  Certificado: auto assinado e dentro do prazo de validade
  Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0 e SSLv3
  No Quadro 12 está o servidor TLS inseguro com as respectivas configurações.

Quadro 12 – Servidor com chave privada de tamanho 1024-bits, algoritmo de *hash* SHA1, com certificado auto assinado e dentro do prazo de validade. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0 e SSLv3

	Servidor
1	mail.ueg.edu.br

Fonte: Autoria propria

- Tamanho da chave privada: 1024-bits
  Algoritmo de *hash*: SHA1
  Certificado: não é auto assinado e está dentro do prazo de validade
  Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0 e SSLv3
  No Quadro 13 está o servidor TLS inseguro com as respectivas configurações.

- Tamanho da chave privada: 2048-bits
  Algoritmo de *hash*: SHA256
  Certificado: não é auto assinado e tem prazo de validade excedido (inválido)
  Versão do protocolo: TLSv1.0, SSLv3
  No Quadro 14 está o servidor TLS inseguro com as respectivas configurações.

Quadro 13 – Servidor com chave privada de tamanho 1024-bits, algoritmo de *hash* SHA1, com certificado que não é auto assinado e está dentro do prazo de validade. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0 e SSLv3

	Servidor
1	h200137217203.ufg.br

Fonte: Autoria propria

Quadro 14 – Servidor com chave privada de tamanho 2048-bits, algoritmo de *hash* SHA256, com certificado que não é auto assinado e tem prazo de validade excedido. Versões do protocolo: TLSv1.0, SSLv3

	Servidor
1	mx.jatai.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 2048-bits

  Algoritmo de *hash*: SHA256

  Certificado: auto assinado e dentro do prazo de validade

  Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0

  No Quadro 15 está o servidor TLS inseguro com as respectivas configurações.

Quadro 15 – Servidor com chave privada de tamanho 2048-bits, algoritmo de *hash* SHA256, com certificado auto assinado e está dentro do prazo de validade. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0

	Servidor
1	terra.eee.ufg.br
2	h200137222222.ufg.br

Fonte: Autoria propria

- Tamanho da chave privada: 2048-bits

  Algoritmo de *hash*: SHA256

  Certificado: não é auto assinado e prazo de validade excedeu (inválido)

  Versão do protocolo: TLSv1.2, TLSv1.1,TLSv1.0

  No Quadro 16 está o servidor TLS inseguro com as respectivas configurações.

Quadro 16 – Servidor com chave privada de tamanho 2048-bits, algoritmo de *hash* SHA256, com certificado que não é auto assinado e o prazo de validade excedeu. Versões do protocolo: TLSv1.2, TLSv1.1,TLSv1.0

	Servidor
1	h200137217219.ufg.br

Fonte: Autoria propria

5.2.2 Protocolos seguros

Seguindo os requisitos de segurança para a configuração do TLS, 30 dos servidores testados são considerados seguros, ou seja, utilizam chave assimétrica, protocolos e algoritmo de *hash* fortes e, certificados válidos. Os servidores seguros são aparesentados no Quadro 17.

Quadro 17 – Relação de servidores TLS seguros: utilizam chave privada de tamanho 2048-bits, algoritmo de *hash* SHA256, certificado válido e versões do protocolo TLS

	Servidor	Servidor	Servidor
1	portal.ufg.fibre.org.br	h200137217132.ufg.br	h200137218137.ufg.br
2	marte.ciar.ufg.br	h200137217133.ufg.br	h200137218139.ufg.br
3	h20013720570.ufg.br	h200137217135.ufg.br	h200137218144.ufg.br
4	shib.ufg.br	h200137217156.ufg.br	mail.ufg.br
5	zabbix.ciar.ufg.br	h200137217159.ufg.br	h200137218148.ufg.br
6	zimbra.ciar.ufg.br	redmine.cercomp.ufg.br	ead.ufg.br
7	h20013721748.ufg.br	h200137217196.ufg.br	webmail.grad.ufg.br
8	20013721752.ufg.br	h200137217205.ufg.br	sistemas.ufg.br
9	h20013721754.ufg.br	oscercomp.ufg.br	h20013722185.ufg.br
10	h200137217130.ufg.br	h200137218130.ufg.br	h200137221180.ufg.br

Fonte: Autoria própria

Os servidores seguros utilizam chave privada de 2048-bits, algoritmo de hash SHA256 e suportam os protocolos TLSv1.0, TLSv1.1 e TLSv1.2. Apesar do *horde5.inf.ufg.br*, *artemis.inf.ufg.br*, *mercurio2.ciar.ufg.br*, *hades.inf.ufg.br*, *shib.ufg.br*, *h20013721752.ufg.br*, *h20013721754.ufg.br*, *h200137217205.ufg.br* e *webmail.grad.ufg.br*, usarem apenas ao protocolo TLSv1.0, não podem ser considerados como inseguros, pois conforme discutido no Capítulo 04, até o momento não há nenhuma falha de segurança conhecida. Os protocolos inseguros por utilizarem chaves fracas, algoritmos de *hash* SHA1, certificados inválidos ou inseguros ou, dar suporte à versão do protocolo SSL, estão sujeitos à ataques *man-in-the-midlle*, colisão de *hash* para gerar certificados frauduletos (MORTON; SMITH, 2014) e ao ataque POODLE, este que é específico da versão SSLv3.

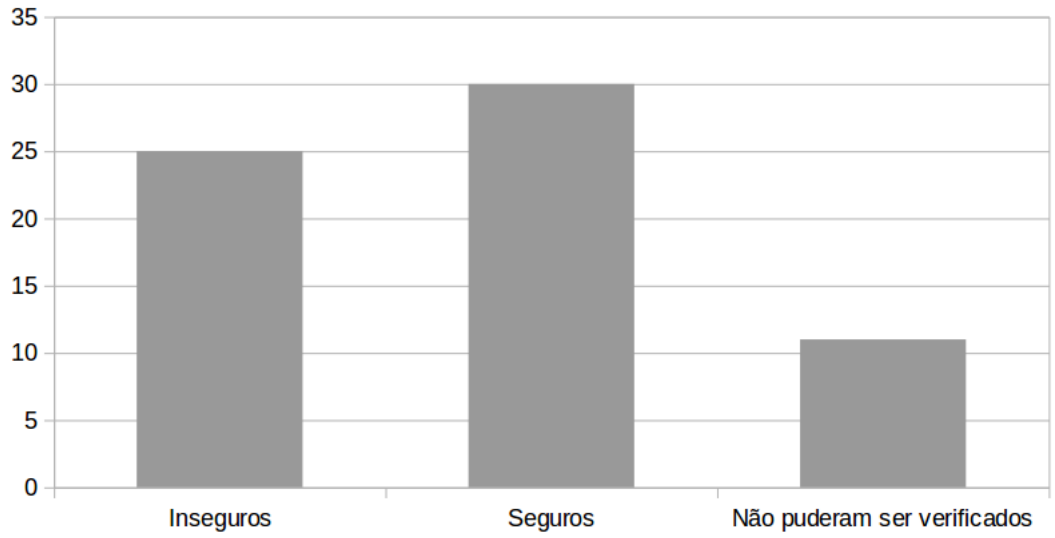
O relatório quanto ao uso seguro do TLS gerado pela ferramenta desenvolvida (*ssl-lufg*) está no Anexo *A.2*, note que há um servidor TLS (*h200137217159.ufg.br*) grifado em vermelho, este foi destacado pois durante os testes realizados o prazo de validade ainda não havia excedido, de maneira que este se enquadrou no grupo de protocolos seguros, por atender aos quatro requisitos de segurança do TLS que foram escolhidos.

5.3 Análise

Na análise de segurança do uso de TLS em estações servidoras na rede da Universidade Federal de Goiás, baseados nos quatro requisitos selecionados, pudemos identificar

que dos 66 servidores TLS, 30 estão configurados da maneira correta, 11 deles não puderam ser verificados devido ao *timeout* e, 25 deles são inseguros e estão suscetíveis à ataques do tipo *man-in-the-middle*, *POODLE* e colisão de *hash*, predominando assim, ainda que com uma diferença pequena, as estações servidoras que configuram o TLS de maneira segura. A representação gráfica desta relação é apresentada na Figura 15.

Figura 15 – Relação dos servidores: seguros, inseguros e que não puderam ser verificados.



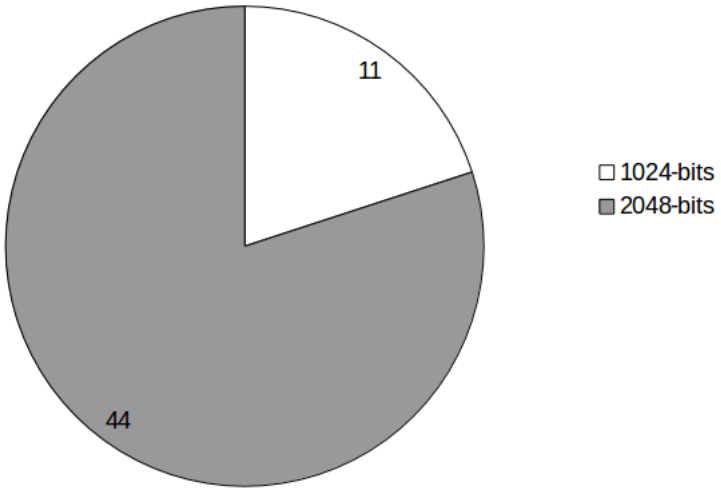
Fonte: Autoria própria

Ao verificar a relação de cada parâmetro, em separado, com todos os servidores analisados, notamos que a maioria utiliza as configurações seguras, mas ao relacionar com outros parâmetros, para 25 dos servidores verificados, temos que ao menos um parâmetro de segurança é considerado inseguro ou inválido. No gráfico da Figura 16 podemos identificar que dos 55 servidores que estabeleceram a conexão TLS, a maioria (44) utitliza chaves assimétricas com tamanho de 2048-bits. A Figura 17 apresenta a relação dos servidores quanto ao algoritmo de *hash* utilizado, dos quais 36 deles já utilizam o SHA256.

Sobre a relação dos servidores TLS com as versões do protocolo que foram utilizadas temos a Figura 18, na qual podemos identificar que a maioria (48) dos servidores utilizam as versões seguras, enquanto apenas 7 deles oferecem suporte à versão insegura e vulnerável ao ataque *POODLE*: SSLv3. Os certificados foram avaliados de acordo com a assinatura (se são auto assinados ou não) e o período de validade (se excedeu ao prazo ou não). Podemos observar na Figura 19 que 38 servidores, de 55, são assinados por uma autoridade certificadora reconhecida e confiável e, que 43 servidores estão dentro do prazo de validade, conforme pode ser visto na Figura 20.

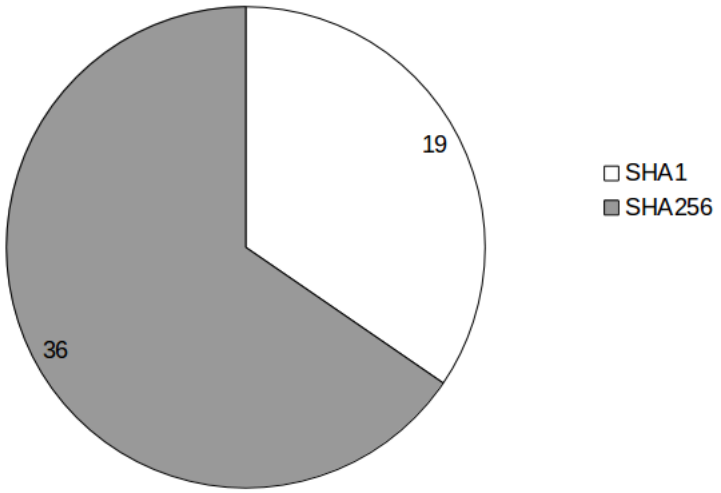
Quando analisamos os quatro parâmetros de segurança para cada servidor TLS, é possível identificar um conjunto de configurações inseguras relacionadas à 25 deles, conforme descrito na Sessão *5.2.1*. A maioria dos servidores inseguros utilizam chaves de 2048-

Figura 16 – Relação dos servidores TLS quanto ao tamanho da chave utilizada.



Fonte: Autoria própria

Figura 17 – Relação dos servidores TLS quanto ao algoritmo de *hash* utilizado.



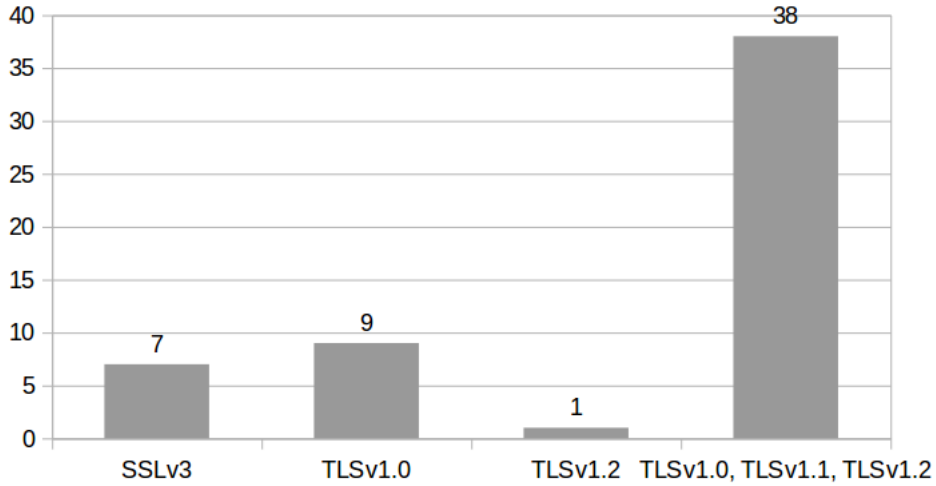
Fonte: Autoria própria

bits, versões seguras do protocolo TLS, possuem certificados dentro do período de validade, mas são auto assinados e ainda utilizam o algoritmo de *hash* SHA1.

Durante a análise também foi possível identificar três servidores com anomalias quanto ao nome de domínio presente nos certificados dos servidores TLS, estes são apresentados no Quadro 19. Os servidores que apresentam anomalias quanto ao nome de domínio possuem as seguintes configurações:

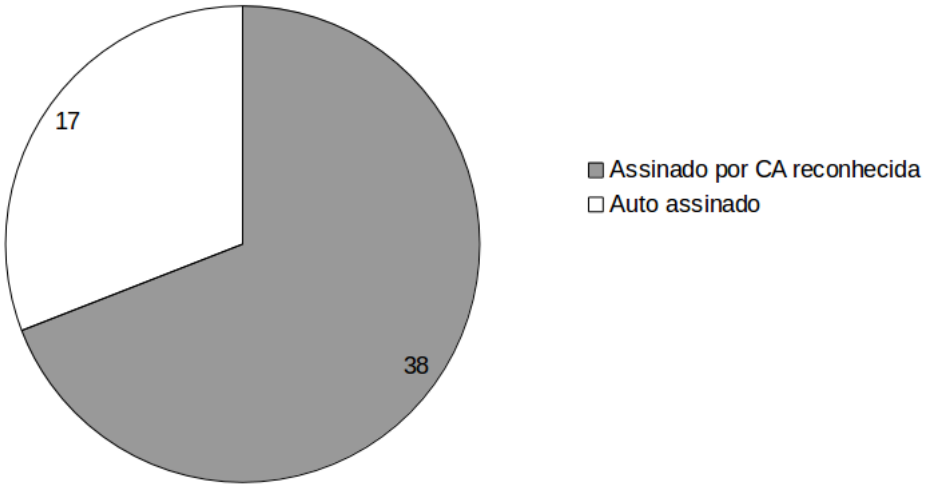
- *h20013721675.ufg.br*: chaves com tamanho 2048-bits, algoritmo de *hash* SHA256, usa as versões de protocolo: SSLv3, TLSv1.0, TLSv1.1 e TLSv1.2, é auto assinado e o período de validade do certificado já expirou.

Figura 18 – Relação dos servidores TLS quanto às versões do protocolo que são utilizadas.



Fonte: Autoria própria

Figura 19 – Relação dos servidores TLS quanto à assinatura dos certificados.

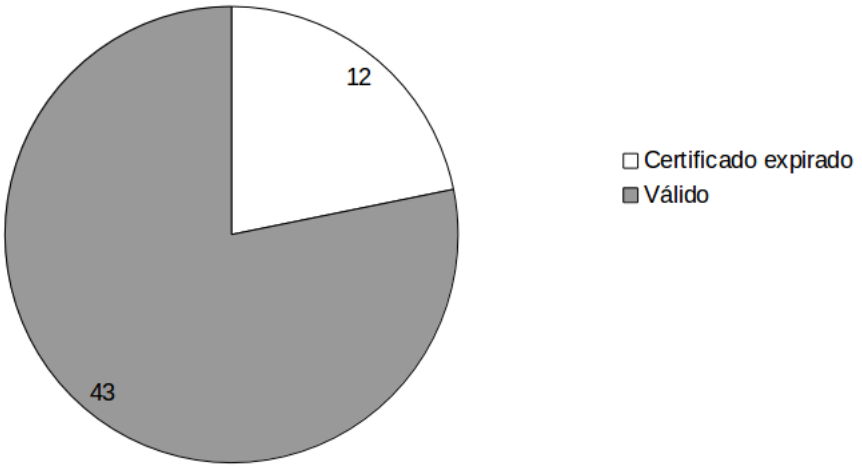


Fonte: Autoria própria

- *h20013721677.ufg.br* e *h20013721678.ufg.br*: chaves com tamanho 2048-bits, algoritmo de *hash* SHA1, usa as versões de protocolo: SSLv3, TLSv1.0, TLSv1.1 e TLSv1.2, é auto assinado e o período de validade do certificado já expirou.

De acordo com os parâmetros de segurança que estes servidores TLS foram configurados e conforme discutido no Capítulo 04, os servidores que possuem chaves menores que 2048-bits, algoritmo de *hash* SHA1, certificado inseguro ou inválido e utilizam a versão de protocolo SSLv3, estão sucetíveis à ataques *man-in-the-middle*, colisão de *hash* e ao ataque *POODLE*, portanto, as anomalias presente no nome de domínio dos servidores TLS citados no Quadro 1, podem representar o fruto de ataques, mas a avaliação destas irregularidades não será tratada neste trabalho, ficando como sugestão para trabalhos futuros.

Figura 20 – Relação dos servidores TLS quanto ao período de validade dos certificados.



Fonte: Autoria própria

Quadro 18 – Relação entre servidores TLS e seus respectivos nome de domínio (CN) com irregularidades

	Servidor	Nome de domínio
1	h20013721675.ufg.br	kett.galo.eco.br
2	h20013721677.ufg.br	DENNA.marceline.com
3	h20013721678.ufg.br	DENNA.marceline.com

Fonte: Autoria própria

CAPÍTULO 6

Conclusão

O TLS é utilizado para garantir a segurança das informações transmitidas ponto-a-ponto, pois garante os requisitos de sigilo, integridade e autenticidade ao utilizar algoritmos de criptografia, assinaturas e certificados digitais. Conforme discorrido ao logo do trabalho, não basta apenas usar o TLS, é preciso configurá-lo da maneira correta, pois do contrário as informações trocadas em uma comunicação que deveria estar segura podem ser totalmente comprometidas e ocasionar danos irreparáveis, visto que estão sujeitas à ataques do tipo *man-in-the-midlle* ou ao ataque *POODLE*.

Como proposta de estudo foram verificados os 66 servidores TLS na rede da Universidade Federal de Goiás, dos quais podemos constatar que a maioria deles implementam o TLS seguro de acordo com os parâmetros de segurança escolhidos: tamanho da chave privada, algoritmo de *hash*, versões do protocolo TLS e validade do certificado. Também foram identificadas em três servidores, anomalias relacionadas ao domínio e que podem ser consideradas como fruto de ataques, mas não foi realizada nenhuma avaliação destas, portanto, poderão ser verificadas em trabalhos futuros. Os servidores com os parâmetros de configurações inseguro foram repassados aos responsáveis pela gerência destes, para que tenham conhecimento e possam tomar as providências necessárias à fim de obter um servidor TLS seguro.

Referências

BURNETT, S.; PAINE, S. *Criptografia e Segurança O Guia Oficial RSA*. Rio de Janeiro: Elsevier, 2002. 367 p. Citado na página 30.

CABALLERO, J. *et al. Engineering Secure Software and Systems*. São Paulo: Springer, 2016. 122-131 p. Citado na página 46.

CENTER, I. K. *Uma visão geral do handshake SSL ou TLS*. 2016. Disponível em: <http://www.ibm.com/support/knowledgecenter/pt-br/SSFKSJ_8.0.0/com.ibm.mq.sec.doc/q009930_.htm#q009930___q009930_5>. Acessado em: 21 de jun. 2016. Citado 2 vezes nas páginas 37 e 39.

DIERKS, T.; RESCORLA, E. *The Transport Layer Security (TLS) Protocol Version 1.2*. 2008. Disponível em: <https://tools.ietf.org/html/rfc5246>. Acessado em: 02 de abr. 2015. Citado na página 30.

DIERKS T.; ALLEN, C. *Can you have too much SSL?* 2009. Disponível em: <https://blog.ivanristic.com/2009/07/can-you-have-too-much-ssl.html>. Acessado em: 23 de mar. 2016. Citado na página 45.

FOROUZAN B.A.; FEGAN, S. C. *Protocolo TCP/IP*. 3. ed. Porto Alegre: AMGH, 2008. 127 p. Citado na página 19.

INTERNET, T. *All about SHA1, SHA2 and SHA256 hash algorithms*. 2016. Disponível em: <https://www.tbs-certificates.co.uk/FAQ/en/sha256.html>. Acessado em: 23 de mar. 2016. Citado na página 44.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet - Uma Abordagem Top-Down*. 5. ed. São Paulo: Pearson, 2010. 592 p. Citado na página 26.

MOELLER, B.; LANGLEY, A. *TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks*. 2014. Disponível em: <https://www.ietf.org/archive/id/draft-bmoeller-tls-downgrade-scsv-02.txt>. Acessado em: 05 de mar. 2015. Citado na página 44.

MOLLER, B. *et al. This POODLE Bites: Exploiting The SSL 3.0 Fallback*. 2014. Disponível em: <https://www.openssl.org/~bodo/ssl-poodle.pdf>. Acessado em: 21 de jun. 2016. Citado na página 46.

MORENO, E. D. *et al. Criptografia em Software e Hardware*. São Paulo: Novatec, 2005. 288 p. Citado 5 vezes nas páginas 22, 23, 24, 26 e 27.

MORTON, B.; SMITH, C. *Why We Need to Move to SHA-2*. 2014. Disponível em: <https://casecurity.org/2014/01/30/why-we-need-to-move-to-sha-2/>. Acessado em: 23 de mar. 2016. Citado 2 vezes nas páginas 44 e 57.

NAKAMURA, E. T.; GEUS, P. L. de. *Segurança de Redes em Ambientes Cooperativos*. São Paulo: Novatec, 2007. 488 p. Citado 6 vezes nas páginas 23, 25, 26, 27, 30 e 43.

NASCIMENTO, A. C. do. *Criptografica e Infraestrutura de Chaves Públicas*. 2009. Disponível em: <http://home.ufam.edu.br/regina_silva/CEGSIC/Textos%20Base/Criptografia_e_ICP.pdf>. Acessado em: 12 de jun. 2016. Citado 4 vezes nas páginas 24, 25, 29 e 30.

Network Associates. *An Introduction to Cryptography*. Santa Clara, CA: Network Associates, 1999. Disponível em: <ftp://ftp.pgpi.org/pub/pgp/6.5/docs/english/IntroToCrypto.pdf>. Acessado em: 26 de jun. 2016. Citado na página 38.

RISTIĆ, I. *The TLS Protocol Version 1.0*. 1999. Disponível em: <https://www.ietf.org/rfc/rfc2246.txt>. Acessado em: 03 de mai. 2016. Citado 2 vezes nas páginas 36 e 42.

____. *Bulletproof SSL and TLS*. London: Feisty Duck Digital, 2014. 506 p. Citado 2 vezes nas páginas 19 e 36.

____. *SSL/TLS Deployment Best Practices*. [S.l.]: Qualys SSL Labs, 2014. Disponível em: <https://www.ssllabs.com/downloads/SSL_TLS_Deployment_Best_Practices.pdf>. Acessado em: 20 de out. 2015. Citado 6 vezes nas páginas 19, 36, 42, 43, 44 e 45.

SANTIN, A. O. *et al. Um modelo de autorização e autenticação baseado em redes de confiança para sistemas distribuídos de larga escala*. 2012. Disponível em: <http://tele.sj.ifsc.edu.br/~mello/artigos/SSI2002-santin.pdf>. Acessado em: 06 de ago. 2016. Citado na página 28.

SARKAR, P. G.; FITZGERALD, S. Attacks on ssl a comprehensive study of beast, crime, time, breach, lucky 13 rc4 biases. *iSEC Partners, Inc*, p. 1–8, 2013. Citado na página 46.

STALLINGS, W. *Criptografia e segurança de redes*. 4. ed. São Paulo: Pearson Prentice Hall, 2008. 492 p. Citado 10 vezes nas páginas 19, 21, 23, 28, 31, 35, 36, 40, 41 e 42.

TANENBAUM, A. S. *Segurança de redes*. Rio de Janeiro: Editora Campus, 2002. 912 p. Citado 3 vezes nas páginas 21, 25 e 45.

TRINTA, F. A. M.; MACEDO, R. C. de. *Um estudo sobre criptografia e assinatura digital*. 1998. Disponível em: <http://www.di.ufpe.br/~flash/ais98/cripto/criptografia.htm>. Acessado em: 08 de nov. 2015. Citado 6 vezes nas páginas 21, 22, 23, 24, 25 e 27.

ANEXO A

Relatório de inferência

A.1 Arquivo *.json* gerado pela ferramenta *ssllabs-scan*

Neste anexo é apresentado o arquivo *.json* gerado pela ferramenta *ssllabs-scan*. Tomamos como exemplo a estação servidora *zimbra.ciar.ufg.br*, conforme citado no Capítulo 05. Arquivos como este foram gerados para todas as outras estações servidoras TLS, para que pudessem ser analisadas e avaliadas como seguras ou não.

["criteriaVersion": "2009l"
"endpoints.0.delegation": 1
"endpoints.0.details.cert.altNames.0": "*.ciar.ufg.br"
"endpoints.0.details.cert.altNames.1": "ciar.ufg.br"
"endpoints.0.details.cert.commonNames.0": "*.ciar.ufg.br"
"endpoints.0.details.cert.crlRevocationStatus": 2
"endpoints.0.details.cert.crlURIs.0": "http://crl.globalsign.com/gs/icpedusha2g2.crl"
"endpoints.0.details.cert.issuerLabel": "ICPEdu"
"endpoints.0.details.cert.issuerSubject": "CN=ICPEdu,O=Rede Nacional de Ensino e Pesquisa - RNP,OU=Gerencia de Servicos (GSer),L=Rio de Janeiro,ST=Rio de Janeiro,C=BR"
"endpoints.0.details.cert.issues": 0
"endpoints.0.details.cert.mustStaple": 0
"endpoints.0.details.cert.notAfter": 1551536465000
"endpoints.0.details.cert.notBefore": 1456842065000
"endpoints.0.details.cert.ocspRevocationStatus": 2
"endpoints.0.details.cert.ocspURIs.0": "http://ocsp2.globalsign.com/icpedusha2g2"
"endpoints.0.details.cert.pinSha256": "ayqI0RV1guv52oOBG5sBn9OkOkMw7Y0payviUiYyGjU="
"endpoints.0.details.cert.revocationInfo": 3
"endpoints.0.details.cert.revocationStatus": 2
"endpoints.0.details.cert.sct": false
"endpoints.0.details.cert.sgc": 0
"endpoints.0.details.cert.sha1Hash": "8deb5397adacb145933f685670cbda7a92c23363"
"endpoints.0.details.cert.sigAlg": "SHA256withRSA"
"endpoints.0.details.cert.subject": "CN=*.ciar.ufg.br,O=UNIVERSIDADE FEDERAL DE GOIAS,L=Goiania,ST=GO,C=BR"
"endpoints.0.details.chain.certs.0.crlRevocationStatus": 2
"endpoints.0.details.chain.certs.0.issuerLabel": "ICPEdu"
"endpoints.0.details.chain.certs.0.issuerSubject": "CN=ICPEdu,O=Rede Nacional de Ensino e Pesquisa - RNP,OU=Gerencia de Servicos (GSer),L=Rio de Janeiro,ST=Rio de Janeiro,C=BR"
"endpoints.0.details.chain.certs.0.issues": 0
"endpoints.0.details.chain.certs.0.keyAlg": "RSA"
"endpoints.0.details.chain.certs.0.keySize": 2048
"endpoints.0.details.chain.certs.0.keyStrength": 2048
"endpoints.0.details.chain.certs.0.label": "*.ciar.ufg.br"
"endpoints.0.details.chain.certs.0.notAfter": 1551536465000
"endpoints.0.details.chain.certs.0.notBefore": 1456842065000
"endpoints.0.details.chain.certs.0.ocspRevocationStatus": 2
"endpoints.0.details.chain.certs.0.pinSha256": "ayqI0RV1guv52oOBG5sBn9OkOkMw7Y0payviUiYyGjU="
"endpoints.0.details.chain.certs.0.raw": "-----BEGIN CERTIFICATE-----
MIIFXTCCBEWgAwIBAgISESERDtdzb4vJPM/L3cIqFxxWMA0GCSqGSIb3DQEBCwUAMIGpMQswCQYD
VQQGEwJCUjEXMBUGA1UECBMOUmlvIGRlIEphbmVpcm8xFzAVBgNVBAcTDlJpbyBkZSBKYW5laXJv
MSQwIgYDVQQLExtHZXJlbmNpYSBkZSBTZXJ2aWNvcyAoR1NlcikxMTAvBgNVBAoTKFJlZGUgTmFj
aW9uYWwgZGUgRW5zaW5vIGUgUGVzcXVpc2EgLSBSTlAxDzANBgNVBAMTBklDUEVkdTAeFw0xNjAz

MDExNDIxMDVaFw0xOTAzMDIxNDIxMDVaMGwxCzAJBgNVBAYTAkJSMQswCQYDVQQIEw
JHTzEQMA4G
A1UEBxMHR29pYW5pYTEmMCQGA1UEChMdVU5JVkVSU0lEQURFIEZFREVSQUwgREUgR
09JQVMxFjAU
BgNVBAMMDSouY2lhci51ZmcuYnIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBA
QDMnD/u
T3MWGvjh0BNnk8UaVrJ/VIGSVk5761h0Xdds8q1Js2A5JyZjr7fDYmRjCUGCxUSHNU53LSZMu
KFk
Tw6ZxxwnDwRQmxw2CnJEI/iSNDOHvx8y8EqCchaOpF5fQLLc3UWYiQFDsIr51Yq3aZ4/7raG7lM
j
6Xi2VK+hWyx+knYZ6yCgJuy1o8Wu/1+0yOpwkx27+Nzeb6Rk5HfrZMtKpHxEegea9YS+s2sQigID
e8LyUjgEERCrFaPUWJUA631faaOQ68lLZexWds7mRCDID1nEAbanE0ZXXr1bl+D94c7wpekjN5nl
Q6Eyo6t2arVXN+CUzHWuaO5otT11QdqTAgMBAAGjggG5MIIBtTAOBgNVHQ8BAf8EBAMCBa
AwSQYD
VR0gBEIwQDA+BgZngQwBAgIwNDAyBggrBgEFBQcCARYmaHR0cHM6Ly93d3cuZ2xvYmFsc2lnbi5j
b20vcmVwb3NpdG9yeS8wJQYDVR0RBB4wHIINKi5jaWFyLnVmZy5icoILY2lhci51ZmcuYnIwCQ
YD
VR0TBAIwADAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwPgYDVR0fBDcwNT
AzoDGgL4Yt
aHR0cDovL2NybC5nbG9iYWxzaWduLmNvbS9ncy9pY3BlZHVzaGEyZzIuY3JsMIGGBggrBgEFB
QcB
AQR6MHgwQAYIKwYBBQUHMAKGNGh0dHA6Ly9zZWN1cmUuZ2xvYmFsc2lnbi5jb20vY2FjZ
XJ0L2lj
cGVkdXNoYTJnMi5jcnQwNAYIKwYBBQUHMAGGKGh0dHA6Ly9vY3NwMi5nbG9iYWxzaWdu
LmNvbS9p
Y3BlZHVzaGEyZzIwHQYDVR0OBBYEFAh6LxV7ooFGyhEEAQMYBlRJ7KLuMB8GA1UdIwQY
MBaAFJXw
pIQap1wgNqbFCNdlQgLld2jjMA0GCSqGSIb3DQEBCwUAA4IBAQBe0actChnmP9wQfepw8qjlOa
DR
TDQ1nH0/pDtecLwjum2FM0Zd1AemW3NU6U7VA7a9RSHliImXXRrAh4p14QrkKMpgRtjTKN7rk
yHk
RBQgwpKOOyO8NQZZzO3HTzk88tNLNu4qRsiahyWo4XwgC+4x3Yr/mw0RjL4vo8Mtl6fbu2yc/
+xy
DufPeJA07l7w4tbvD4Rw1tJOAnTKEyg2VIvs8RlEDSPuFUB5c4temTiAe5tWYk9+HJ2P5fKq3JK0
CRjpnjkpTehNADofU6BaBFw6Kl6OylMW3OZ7xWODm9wC+gHlj9sie4MD+9YyaVOhlhAgMGQf
r/gN
xdpVMgwjoB0R
-----END CERTIFICATE-----
"
"endpoints.0.details.chain.certs.0.revocationStatus": 2
"endpoints.0.details.chain.certs.0.sha1Hash": "8deb5397adacb145933f685670cbda7a92c23363"
"endpoints.0.details.chain.certs.0.sigAlg": "SHA256withRSA"
"endpoints.0.details.chain.certs.0.subject": "CN=*.ciar.ufg.br,O=UNIVERSIDADE FEDERAL DE
GOIAS,L=Goiania,ST=GO,C=BR"
"endpoints.0.details.chain.certs.1.crlRevocationStatus": 2
"endpoints.0.details.chain.certs.1.issuerLabel": "Trusted Root CA SHA256 G2"
"endpoints.0.details.chain.certs.1.issuerSubject": "CN=Trusted Root CA SHA256 G2,O=GlobalSign
nv-sa,OU=Trusted Root,C=BE"

"endpoints.0.details.chain.certs.1.issues": 0
"endpoints.0.details.chain.certs.1.keyAlg": "RSA"
"endpoints.0.details.chain.certs.1.keySize": 2048
"endpoints.0.details.chain.certs.1.keyStrength": 2048
"endpoints.0.details.chain.certs.1.label": "ICPEdu"
"endpoints.0.details.chain.certs.1.notAfter": 1614816000000
"endpoints.0.details.chain.certs.1.notBefore": 1393891200000
"endpoints.0.details.chain.certs.1.ocspRevocationStatus": 2
"endpoints.0.details.chain.certs.1.pinSha256":
"m/Zke2f6Q6acou+ZOFg+8OdSCzint/5uosSYaFXE3EM="
"endpoints.0.details.chain.certs.1.raw": "-----BEGIN CERTIFICATE-----
MIIFHjCCBAagAwIBAgIRV7Ce72FWEIdEkeksVGL0YZYwDQYJKoZIhvcNAQELBQAwYzELM
AkGA1UE
BhMCQkUxFTATBgNVBAsTDFRydXN0ZWQgUm9vdDEZMBcGA1UEChMQR2xvYmFsU2lnbiB
udi1zYTEi
MCAGA1UEAxMZVHJ1c3RlZCBSb290IENBIFNIQTI1NiBHMjAeFw0xNDAzMDQwMDAwMDB
aFw0yMTAz
MDQwMDAwMDBaMIGpMQswCQYDVQQGEwJCUjEXMBUGA1UECBMOUmlvIGRlIEphbmV
pcm8xFzAVBgNV
BAcTDlJpbyBkZSBKYW5laXJvMSQwIgYDVQQLExtHZXJlbmNpYSBkZSBTZXJ2aWNvcyAoR1
Nlcikx
MTAvBgNVBAoTKFJlZGUgTmFjaW9uYWwgZGUgRW5zaW5vIGUgUGVzcXVpc2EgLSBSTlAx
DzANBgNV
BAMTBklDUEVkdTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKBD/AYytLrw
wqh3AxTw
yumXpiEgdo9356XAhOXTrzUgtvtjkqgOkUDkMFzEWWEzPV1EAelH4csW93DYXVO+C5kpWF+j
W3gj
6Vu8HNkouiKM5BdmZWp338h6db+rSPMDB0IOc1DMOQB7gtNL3awXOR1yKr2jP5h+LboGDxL
GfwdY
vONTkmr7nTbx+cR3Kj8Je1MbnwkMt1xLpk21Yo59RV46zLaY+JUMHd8Z7fgVBIcDx/ASdoTR3e1
S
hIWghUowlOdK7vtdo3lZPfDlXsq8rjPUDPudk/CVjnW1b6pKSB6ez8+nmSiw87brmkUCV105HiE7
J/Gc2cifblIKn4XPqncCAwEAAaOCAYQwggGAMA4GA1UdDwEB/wQEAwIBBjBHBgNVHSAEQ
DA+MDwG
BFUdIAAwNDAyBggrBgEFBQcCARYmaHR0cHM6Ly93d3cuZ2xvYmFsc2lnbi5jb20vcmVwb3Npd
G9y
eS8wEgYDVR0TAQH/BAgwBgEB/wIBADBBBgNVHR8EOjA4MDagNKAyhjBodHRwOi8vY3JsL
mdsb2Jh
bHNpZ24uY29tL2dzL3RydXN0cm9vdHNoYTJnMi5jcmwwgY0GCCsGAQUFBwEBBIGAMH4wN
wYIKwYB
BQUHMAGGK2h0dHA6Ly9vY3NwMi5nbG9iYWxzaWduLmNvbS90cnVzdHJvb3RzaGEyZzIwQw
YIKwYB
BQUHMAKGN2h0dHA6Ly9zZWN1cmUuZ2xvYmFsc2lnbi5jb20vY2FjZXJ0L3RydXN0cm9vdHNo
YTJn
Mi5jcnQwHQYDVR0OBBYEFJXwpIQap1wgNqbFCNdlQgLld2jjMB8GA1UdIwQYMBaAFMhjmw
hpVMKY
yNnN4zO3UF74yQGbMA0GCSqGSIb3DQEBCwUAA4IBAQBk3q+/7S1BpzW83NDHdXF8Jx0/+E
3tqnyl
5MphjGh9oKk3I8hCzPQgetE03awPQhDRf6r64SnIkhr8/dWm9edpsBJFj7PwXzlfq6zI6S0hIov/

A+59lLeCVDPTIOu3m8CHEQeKtoesWUXgrYEY5F0fl/E8V5SQ5NKrDC88ipNWerMKhYEZu1HN
+fSt
JLWv/H3v0x1hRVElrxdolHwh2ANhkFVquUALQihDN5Q5DwO/5jcvqm/XYK2fGQAStBGG/NgongTi
bMpHGOq62jv1HcnHhpVUfZMop0KX4i60puToNn9kIW8KzclGLd7EKypOPkSbcBmQzwivcyo7HkPU
Dg9o
-----END CERTIFICATE-----
"
"endpoints.0.details.chain.certs.1.revocationStatus": 2
"endpoints.0.details.chain.certs.1.sha1Hash": "4731fc3e37f4f49949739acd831f562dd8bcabdc"
"endpoints.0.details.chain.certs.1.sigAlg": "SHA256withRSA"
"endpoints.0.details.chain.certs.1.subject": "CN=ICPEdu,O=Rede Nacional de Ensino e Pesquisa - RNP,OU=Gerencia de Servicos (GSer),L=Rio de Janeiro,ST=Rio de Janeiro,C=BR"
"endpoints.0.details.chain.certs.2.crlRevocationStatus": 2
"endpoints.0.details.chain.certs.2.issuerLabel": "GlobalSign"
"endpoints.0.details.chain.certs.2.issuerSubject": "CN=GlobalSign,O=GlobalSign,OU=GlobalSign Root CA - R3"
"endpoints.0.details.chain.certs.2.issues": 0
"endpoints.0.details.chain.certs.2.keyAlg": "RSA"
"endpoints.0.details.chain.certs.2.keySize": 2048
"endpoints.0.details.chain.certs.2.keyStrength": 2048
"endpoints.0.details.chain.certs.2.label": "Trusted Root CA SHA256 G2"
"endpoints.0.details.chain.certs.2.notAfter": 1808650800000
"endpoints.0.details.chain.certs.2.notBefore": 1335351600000
"endpoints.0.details.chain.certs.2.ocspRevocationStatus": 2
"endpoints.0.details.chain.certs.2.pinSha256": "HHWscHR+mXReMKBRZxCvqEg6wDv6HAbPzKN7NlLvq4c="
"endpoints.0.details.chain.certs.2.raw": "-----BEGIN CERTIFICATE-----
MIIEXDCCA0SgAwIBAgILBAAAAAABNumCOV0wDQYJKoZIhvcNAQELBQAwTDEgMB4GA1UECxMXR2xv
YmFsU2lnbiBSb290IENBIC0gUjMxEzARBgNVBAoTCkdsb2JhbFNpZ24xEzARBgNVBAMTCkdsb2Jh
bFNpZ24wHhcNMTIwNDI1MTEwMDAwWhcNMjcwNDI1MTEwMDAwWjBjMQswCQYDVQQGEwJCRTEVMBMG
A1UECxMMVHJ1c3RlZCBSb290MRkwFwYDVQQKExBHbG9iYWxTaWduIG52LXNhMSIwIAYDVQQDExlU
cnVzdGVkIFJvb3QgQ0EgU0hBMjU2IEcyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
z80+/Q2PAhLuYwe04YTLBLGKr1/JScHtDvAY5E94GjGxCbSR1/1VhL880UPJyN85tddOoxZPgtIy
ZixDvvK+CgpT5webyBBbqK/ap7aoByghAJ7X520XZMRwKA6cEWa6tjCLWH1zscxQxGzgtV50rn2u
x2SapoCPxMpM4+tpEVwWJf3KP3NT+jd9GRaXWgNei5JKQuo9l+cZkSeuoWijvaer5hcLCufPywMM
Qd0r6XXIM/l7g9DjMaE24d+fa2bWxQXC8WT/PZ+D1KUEkdtn/ixADqsoiIibGn7M84EE9/NLjbzP
rwROlBUJFz6cuw+II0rZ8OFFeZ/OkHHYZq2h9wIDAQABo4IBJjCCASIwDgYDVR0PAQH/BAQDAgEG
MA8GA1UdEwEB/wQFMAMBAf8wRwYDVR0gBEAwPjA8BgRVHSAAMDQwMgYIKwYBBQUHAgEWJmh0dHBz

Oi8vd3d3Lmdsb2JhbHNpZ24uY29tL3JlcG9zaXRvcnkvMB0GA1UdDgQWBBTIY5sIaVTCmMjZze
Mz
t1Be+MkBmzA2BgNVHR8ELzAtMCugKaAnhiVodHRwOi8vY3JsLmdsb2JhbHNpZ24ubmV0L3Jvb
3Qt
cjMuY3JsMD4GCCsGAQUFBwEBBDIwMDAuBggrBgEFBQcwAYYiaHR0cDovL29jc3AyLmdsb2J
hbHNp
Z24uY29tL3Jvb3RyMzAfBgNVHSMEGDAWgBSP8Et/qC5FJK5NUPpjmove4t0bvDANBgkqhkiG9
w0B
AQsFAAOCAQEAXzbLwBjJiY6j3WEcxD3eVnsIY4pY3bl6660tgpxCuLVx4o1xyiVkS/BcQFD7GIo
X
FBRrf5HibO1uSEOw0QZoRwlsio1VPg1PRaccG5C1sB51l/TL1XH5zldZBCnRYrrFqCPorxi0xoRo
gj8kqkS2xyzYLElhx9X7jIzfZ8dC4mgOeoCtVvwM9xvmef3n6Vyb7/hl3w/zWwKxWyKJNaF7tScD
5nvtLUzyBpr++aztiyJ1WliWcS6W+V2gKg9rxEC/rc2yJS70DvfkPiEnBJ2x2AHZV3yKTALUqurk
V705JledqUT9I5frAwYNXZ8pNzden+DIcSIo7yKy6MX9czbFWQ==
-----END CERTIFICATE-----
"
"endpoints.0.details.chain.certs.2.revocationStatus": 2
"endpoints.0.details.chain.certs.2.sha1Hash": "9abb55a26f9c06d500c45991f02c15b55d00a702"
"endpoints.0.details.chain.certs.2.sigAlg": "SHA256withRSA"
"endpoints.0.details.chain.certs.2.subject": "CN=Trusted Root CA SHA256 G2,O=GlobalSign nv-
sa,OU=Trusted Root,C=BE"
"endpoints.0.details.chain.certs.3.crlRevocationStatus": 0
"endpoints.0.details.chain.certs.3.issuerLabel": "GlobalSign"
"endpoints.0.details.chain.certs.3.issuerSubject": "CN=GlobalSign,O=GlobalSign,OU=GlobalSign
Root CA - R3"
"endpoints.0.details.chain.certs.3.issues": 0
"endpoints.0.details.chain.certs.3.keyAlg": "RSA"
"endpoints.0.details.chain.certs.3.keySize": 2048
"endpoints.0.details.chain.certs.3.keyStrength": 2048
"endpoints.0.details.chain.certs.3.label": "GlobalSign"
"endpoints.0.details.chain.certs.3.notAfter": 1868522400000
"endpoints.0.details.chain.certs.3.notBefore": 1237370400000
"endpoints.0.details.chain.certs.3.ocspRevocationStatus": 0
"endpoints.0.details.chain.certs.3.pinSha256":
"cGuxAXyFXFkWm61cF4HPWX8S0srS9j0aSqN0k4AP+4A="
"endpoints.0.details.chain.certs.3.raw": "-----BEGIN CERTIFICATE-----
MIIDXzCCAkegAwIBAgILBAAAAAABIVhTCKIwDQYJKoZIhvcNAQELBQAwTDEgMB4GA1U
ECxMXR2xv
YmFsU2lnbiBSb290IENBIC0gUjMxEzARBgNVBAoTCkdsb2JhbFNpZ24xEzARBgNVBAMTCkds
b2Jh
bFNpZ24wHhcNMDkwMzE4MTAwMDAwWhcNMjkwMzE4MTAwMDAwWjBMMSAwHgYDVQ
QLExdHbG9iYWxT
aWduIFJvb3QgQ0EgLSBSMzETMBEGA1UEChMKR2xvYmFsU2lnbjETMBEGA1UEAxMKR2xv
YmFsU2ln
bjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMwldpB5BngiFvXAg7aEyiie/QV
2EcWt
iHL8RgJDx7KKnQRfJMsuS+FggkbhUqsMgUdwbN1k0ev1LKMPgj0MK66X17YUhhB5uzsTgHeM
COFJ
0mpiLx9e+pZo34knlTifBtc+ycsmWQ1z3rDI6SYOgxXG71uL0gRgykmmKPZpO/bLyCiR5Z2KYVc3

rHQU3HTgOu5yLy6c+9C7v/U9AOEGM+iCK65TpjoWc4zdQQ4gOsC0p6Hpsk+QLjJg6VfLuQSSaG
jl
OCZgdbKfd/
+RFO+uIEn8rUAVSNECMWEZXriX7613t2Saer9fwRPvm2L7DWzgVGkWqQPabumDk3F2
xmmFghcCAwEAAaNCMEAwDgYDVR0PAQH/BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8w
HQYDVR0OBBYE
FI/wS3+oLkUkrk1Q+mOai97i3Ru8MA0GCSqGSIb3DQEBCwUAA4IBAQBLQNvAUKr+yAzv95Z
URUm7
lgAJQayzE4aGKAczymvmdLm6AC2upArT9fHxD4q/c2dKg8dEe3jgr25sbwMpjjM5RcOO5LlXbKr8
EpbsU8Yt5CRsuZRj+9xTaGdWPoO4zzUhw8lo/s7awlOqzJCK6fBdRoyV3XpYKBovHd7NADdBj+1
E
bddTKJd+82cEHhXXipa0095MJ6RMG3NzdvQXmcIfeg7jLQitChws/zyrVQ4PkX4268NXSb7hLi18
YIvDQVETI53O9zJrlAGomecsMx86OyXShkDOOyyGeMlhLxS67ttVb9+E7gUJTb0o2HLO02JQZR
7r
kpeDMdmztcpHWD9f
-----END CERTIFICATE-----
"
"endpoints.0.details.chain.certs.3.revocationStatus": 0
"endpoints.0.details.chain.certs.3.sha1Hash": "d69b561148f01c77c54578c10926df5b856976ad"
"endpoints.0.details.chain.certs.3.sigAlg": "SHA256withRSA"
"endpoints.0.details.chain.certs.3.subject": "CN=GlobalSign,O=GlobalSign,OU=GlobalSign Root CA
- R3"
"endpoints.0.details.chain.issues": 16
"endpoints.0.details.compressionMethods": 0
"endpoints.0.details.dhPrimes.0":
"fd7f53811d75122952df4a9c2eece4e7f611b7523cef4400c31e3f80b6512669455d402251fb593d8d58fa
bfc5f5ba30f6cb9b556cd7813b801d346ff26660b76b9950a5a49f9fe8047b1022c24fbba9d7feb7c61bf83
b57e7c6a8a6150f04fb83f6d3c51ec3023554135a169132f675f3ae2b61d72aeff22203199dd14801c7"
"endpoints.0.details.dhUsesKnownPrimes": 2
"endpoints.0.details.dhYsReuse": false
"endpoints.0.details.drownErrors": false
"endpoints.0.details.drownVulnerable": false
"endpoints.0.details.fallbackScsv": false
"endpoints.0.details.forwardSecrecy": 1
"endpoints.0.details.freak": false
"endpoints.0.details.hasSct": 0
"endpoints.0.details.heartbeat": false
"endpoints.0.details.heartbleed": false
"endpoints.0.details.hostStartTime": 1470370429172
"endpoints.0.details.hpkpPolicy.status": "absent"
"endpoints.0.details.hpkpRoPolicy.status": "absent"
"endpoints.0.details.hstsPolicy.LONG_MAX_AGE": 15552000
"endpoints.0.details.hstsPolicy.status": "absent"
"endpoints.0.details.hstsPreloads.0.hostname": "zimbra.ciar.ufg.br"
"endpoints.0.details.hstsPreloads.0.source": "Chrome"
"endpoints.0.details.hstsPreloads.0.sourceTime": 1470370500775
"endpoints.0.details.hstsPreloads.0.status": "absent"
"endpoints.0.details.hstsPreloads.1.hostname": "zimbra.ciar.ufg.br"
"endpoints.0.details.hstsPreloads.1.source": "Edge"

"endpoints.0.details.hstsPreloads.1.sourceTime": 1470370440864
"endpoints.0.details.hstsPreloads.1.status": "absent"
"endpoints.0.details.hstsPreloads.2.hostname": "zimbra.ciar.ufg.br"
"endpoints.0.details.hstsPreloads.2.source": "Firefox"
"endpoints.0.details.hstsPreloads.2.sourceTime": 1470370440864
"endpoints.0.details.hstsPreloads.2.status": "absent"
"endpoints.0.details.hstsPreloads.3.hostname": "zimbra.ciar.ufg.br"
"endpoints.0.details.hstsPreloads.3.source": "IE"
"endpoints.0.details.hstsPreloads.3.sourceTime": 1470370440864
"endpoints.0.details.hstsPreloads.3.status": "absent"
"endpoints.0.details.hstsPreloads.4.hostname": "zimbra.ciar.ufg.br"
"endpoints.0.details.hstsPreloads.4.source": "Tor"
"endpoints.0.details.hstsPreloads.4.sourceTime": 1470154501314
"endpoints.0.details.hstsPreloads.4.status": "absent"
"endpoints.0.details.httpStatusCode": 200
"endpoints.0.details.key.alg": "RSA"
"endpoints.0.details.key.debianFlaw": false
"endpoints.0.details.key.size": 2048
"endpoints.0.details.key.strength": 2048
"endpoints.0.details.logjam": false
"endpoints.0.details.nonPrefixDelegation": true
"endpoints.0.details.ocspStapling": false
"endpoints.0.details.openSSLLuckyMinus20": 1
"endpoints.0.details.openSslCcs": 1
"endpoints.0.details.poodle": false
"endpoints.0.details.poodleTls": 1
"endpoints.0.details.prefixDelegation": false
"endpoints.0.details.protocols.0.id": 769
"endpoints.0.details.protocols.0.name": "TLS"
"endpoints.0.details.protocols.0.version": "1.0"
"endpoints.0.details.protocols.1.id": 770
"endpoints.0.details.protocols.1.name": "TLS"
"endpoints.0.details.protocols.1.version": "1.1"
"endpoints.0.details.protocols.2.id": 771
"endpoints.0.details.protocols.2.name": "TLS"
"endpoints.0.details.protocols.2.version": "1.2"
"endpoints.0.details.rc4Only": false
"endpoints.0.details.rc4WithModern": false
"endpoints.0.details.renegSupport": 6
"endpoints.0.details.sessionResumption": 2
"endpoints.0.details.sessionTickets": 0
"endpoints.0.details.sims.results.0.attempts": 1
"endpoints.0.details.sims.results.0.client.id": 56
"endpoints.0.details.sims.results.0.client.isReference": false
"endpoints.0.details.sims.results.0.client.name": "Android"
"endpoints.0.details.sims.results.0.client.version": "2.3.7"
"endpoints.0.details.sims.results.0.errorCode": 0
"endpoints.0.details.sims.results.0.protocolId": 769
"endpoints.0.details.sims.results.0.suiteId": 4

"endpoints.0.details.sims.results.1.attempts": 1
"endpoints.0.details.sims.results.1.client.id": 58
"endpoints.0.details.sims.results.1.client.isReference": false
"endpoints.0.details.sims.results.1.client.name": "Android"
"endpoints.0.details.sims.results.1.client.version": "4.0.4"
"endpoints.0.details.sims.results.1.errorCode": 0
"endpoints.0.details.sims.results.1.protocolId": 769
"endpoints.0.details.sims.results.1.suiteId": 49170
"endpoints.0.details.sims.results.10.attempts": 1
"endpoints.0.details.sims.results.10.client.id": 126
"endpoints.0.details.sims.results.10.client.isReference": true
"endpoints.0.details.sims.results.10.client.name": "Chrome"
"endpoints.0.details.sims.results.10.client.platform": "Win 7"
"endpoints.0.details.sims.results.10.client.version": "51"
"endpoints.0.details.sims.results.10.errorCode": 0
"endpoints.0.details.sims.results.10.protocolId": 771
"endpoints.0.details.sims.results.10.suiteId": 49199
"endpoints.0.details.sims.results.11.attempts": 1
"endpoints.0.details.sims.results.11.client.id": 84
"endpoints.0.details.sims.results.11.client.isReference": false
"endpoints.0.details.sims.results.11.client.name": "Firefox"
"endpoints.0.details.sims.results.11.client.platform": "Win 7"
"endpoints.0.details.sims.results.11.client.version": "31.3.0 ESR"
"endpoints.0.details.sims.results.11.errorCode": 0
"endpoints.0.details.sims.results.11.protocolId": 771
"endpoints.0.details.sims.results.11.suiteId": 49199
"endpoints.0.details.sims.results.12.attempts": 1
"endpoints.0.details.sims.results.12.client.id": 128
"endpoints.0.details.sims.results.12.client.isReference": true
"endpoints.0.details.sims.results.12.client.name": "Firefox"
"endpoints.0.details.sims.results.12.client.platform": "Win 7"
"endpoints.0.details.sims.results.12.client.version": "46"
"endpoints.0.details.sims.results.12.errorCode": 0
"endpoints.0.details.sims.results.12.protocolId": 771
"endpoints.0.details.sims.results.12.suiteId": 49199
"endpoints.0.details.sims.results.13.attempts": 1
"endpoints.0.details.sims.results.13.client.id": 132
"endpoints.0.details.sims.results.13.client.isReference": true
"endpoints.0.details.sims.results.13.client.name": "Firefox"
"endpoints.0.details.sims.results.13.client.platform": "Win 7"
"endpoints.0.details.sims.results.13.client.version": "47"
"endpoints.0.details.sims.results.13.errorCode": 0
"endpoints.0.details.sims.results.13.protocolId": 771
"endpoints.0.details.sims.results.13.suiteId": 49199
"endpoints.0.details.sims.results.14.attempts": 1
"endpoints.0.details.sims.results.14.client.id": 97
"endpoints.0.details.sims.results.14.client.isReference": false
"endpoints.0.details.sims.results.14.client.name": "Googlebot"
"endpoints.0.details.sims.results.14.client.version": "Feb 2015"

"endpoints.0.details.sims.results.14.errorCode": 0
"endpoints.0.details.sims.results.14.protocolId": 771
"endpoints.0.details.sims.results.14.suiteId": 49199
"endpoints.0.details.sims.results.15.attempts": 1
"endpoints.0.details.sims.results.15.client.id": 100
"endpoints.0.details.sims.results.15.client.isReference": false
"endpoints.0.details.sims.results.15.client.name": "IE"
"endpoints.0.details.sims.results.15.client.platform": "XP"
"endpoints.0.details.sims.results.15.client.version": "6"
"endpoints.0.details.sims.results.15.errorCode": 1
"endpoints.0.details.sims.results.16.attempts": 1
"endpoints.0.details.sims.results.16.client.id": 19
"endpoints.0.details.sims.results.16.client.isReference": false
"endpoints.0.details.sims.results.16.client.name": "IE"
"endpoints.0.details.sims.results.16.client.platform": "Vista"
"endpoints.0.details.sims.results.16.client.version": "7"
"endpoints.0.details.sims.results.16.errorCode": 0
"endpoints.0.details.sims.results.16.protocolId": 769
"endpoints.0.details.sims.results.16.suiteId": 47
"endpoints.0.details.sims.results.17.attempts": 1
"endpoints.0.details.sims.results.17.client.id": 101
"endpoints.0.details.sims.results.17.client.isReference": false
"endpoints.0.details.sims.results.17.client.name": "IE"
"endpoints.0.details.sims.results.17.client.platform": "XP"
"endpoints.0.details.sims.results.17.client.version": "8"
"endpoints.0.details.sims.results.17.errorCode": 0
"endpoints.0.details.sims.results.17.protocolId": 769
"endpoints.0.details.sims.results.17.suiteId": 4
"endpoints.0.details.sims.results.18.attempts": 1
"endpoints.0.details.sims.results.18.client.id": 113
"endpoints.0.details.sims.results.18.client.isReference": true
"endpoints.0.details.sims.results.18.client.name": "IE"
"endpoints.0.details.sims.results.18.client.platform": "Win 7"
"endpoints.0.details.sims.results.18.client.version": "8-10"
"endpoints.0.details.sims.results.18.errorCode": 0
"endpoints.0.details.sims.results.18.protocolId": 769
"endpoints.0.details.sims.results.18.suiteId": 49171
"endpoints.0.details.sims.results.19.attempts": 1
"endpoints.0.details.sims.results.19.client.id": 133
"endpoints.0.details.sims.results.19.client.isReference": true
"endpoints.0.details.sims.results.19.client.name": "IE"
"endpoints.0.details.sims.results.19.client.platform": "Win 7"
"endpoints.0.details.sims.results.19.client.version": "11"
"endpoints.0.details.sims.results.19.errorCode": 0
"endpoints.0.details.sims.results.19.protocolId": 771
"endpoints.0.details.sims.results.19.suiteId": 49191
"endpoints.0.details.sims.results.2.attempts": 1
"endpoints.0.details.sims.results.2.client.id": 59
"endpoints.0.details.sims.results.2.client.isReference": false

"endpoints.0.details.sims.results.2.client.name": "Android"
"endpoints.0.details.sims.results.2.client.version": "4.1.1"
"endpoints.0.details.sims.results.2.errorCode": 0
"endpoints.0.details.sims.results.2.protocolId": 769
"endpoints.0.details.sims.results.2.suiteId": 49170
"endpoints.0.details.sims.results.20.attempts": 1
"endpoints.0.details.sims.results.20.client.id": 134
"endpoints.0.details.sims.results.20.client.isReference": true
"endpoints.0.details.sims.results.20.client.name": "IE"
"endpoints.0.details.sims.results.20.client.platform": "Win 8.1"
"endpoints.0.details.sims.results.20.client.version": "11"
"endpoints.0.details.sims.results.20.errorCode": 0
"endpoints.0.details.sims.results.20.protocolId": 771
"endpoints.0.details.sims.results.20.suiteId": 49191
"endpoints.0.details.sims.results.21.attempts": 1
"endpoints.0.details.sims.results.21.client.id": 64
"endpoints.0.details.sims.results.21.client.isReference": false
"endpoints.0.details.sims.results.21.client.name": "IE"
"endpoints.0.details.sims.results.21.client.platform": "Win Phone 8.0"
"endpoints.0.details.sims.results.21.client.version": "10"
"endpoints.0.details.sims.results.21.errorCode": 0
"endpoints.0.details.sims.results.21.protocolId": 769
"endpoints.0.details.sims.results.21.suiteId": 47
"endpoints.0.details.sims.results.22.attempts": 1
"endpoints.0.details.sims.results.22.client.id": 65
"endpoints.0.details.sims.results.22.client.isReference": true
"endpoints.0.details.sims.results.22.client.name": "IE"
"endpoints.0.details.sims.results.22.client.platform": "Win Phone 8.1"
"endpoints.0.details.sims.results.22.client.version": "11"
"endpoints.0.details.sims.results.22.errorCode": 0
"endpoints.0.details.sims.results.22.protocolId": 771
"endpoints.0.details.sims.results.22.suiteId": 60
"endpoints.0.details.sims.results.23.attempts": 1
"endpoints.0.details.sims.results.23.client.id": 106
"endpoints.0.details.sims.results.23.client.isReference": true
"endpoints.0.details.sims.results.23.client.name": "IE"
"endpoints.0.details.sims.results.23.client.platform": "Win Phone 8.1 Update"
"endpoints.0.details.sims.results.23.client.version": "11"
"endpoints.0.details.sims.results.23.errorCode": 0
"endpoints.0.details.sims.results.23.protocolId": 771
"endpoints.0.details.sims.results.23.suiteId": 49191
"endpoints.0.details.sims.results.24.attempts": 1
"endpoints.0.details.sims.results.24.client.id": 131
"endpoints.0.details.sims.results.24.client.isReference": true
"endpoints.0.details.sims.results.24.client.name": "IE"
"endpoints.0.details.sims.results.24.client.platform": "Win 10"
"endpoints.0.details.sims.results.24.client.version": "11"
"endpoints.0.details.sims.results.24.errorCode": 0
"endpoints.0.details.sims.results.24.protocolId": 771

"endpoints.0.details.sims.results.24.suiteId": 49199
"endpoints.0.details.sims.results.25.attempts": 1
"endpoints.0.details.sims.results.25.client.id": 130
"endpoints.0.details.sims.results.25.client.isReference": true
"endpoints.0.details.sims.results.25.client.name": "Edge"
"endpoints.0.details.sims.results.25.client.platform": "Win 10"
"endpoints.0.details.sims.results.25.client.version": "13"
"endpoints.0.details.sims.results.25.errorCode": 0
"endpoints.0.details.sims.results.25.protocolId": 771
"endpoints.0.details.sims.results.25.suiteId": 49199
"endpoints.0.details.sims.results.26.attempts": 1
"endpoints.0.details.sims.results.26.client.id": 120
"endpoints.0.details.sims.results.26.client.isReference": true
"endpoints.0.details.sims.results.26.client.name": "Edge"
"endpoints.0.details.sims.results.26.client.platform": "Win Phone 10"
"endpoints.0.details.sims.results.26.client.version": "13"
"endpoints.0.details.sims.results.26.errorCode": 0
"endpoints.0.details.sims.results.26.protocolId": 771
"endpoints.0.details.sims.results.26.suiteId": 49199
"endpoints.0.details.sims.results.27.attempts": 1
"endpoints.0.details.sims.results.27.client.id": 25
"endpoints.0.details.sims.results.27.client.isReference": false
"endpoints.0.details.sims.results.27.client.name": "Java"
"endpoints.0.details.sims.results.27.client.version": "6u45"
"endpoints.0.details.sims.results.27.errorCode": 1
"endpoints.0.details.sims.results.28.attempts": 1
"endpoints.0.details.sims.results.28.client.id": 26
"endpoints.0.details.sims.results.28.client.isReference": false
"endpoints.0.details.sims.results.28.client.name": "Java"
"endpoints.0.details.sims.results.28.client.version": "7u25"
"endpoints.0.details.sims.results.28.errorCode": 0
"endpoints.0.details.sims.results.28.protocolId": 769
"endpoints.0.details.sims.results.28.suiteId": 49171
"endpoints.0.details.sims.results.29.attempts": 1
"endpoints.0.details.sims.results.29.client.id": 86
"endpoints.0.details.sims.results.29.client.isReference": false
"endpoints.0.details.sims.results.29.client.name": "Java"
"endpoints.0.details.sims.results.29.client.version": "8u31"
"endpoints.0.details.sims.results.29.errorCode": 0
"endpoints.0.details.sims.results.29.protocolId": 771
"endpoints.0.details.sims.results.29.suiteId": 49191
"endpoints.0.details.sims.results.3.attempts": 1
"endpoints.0.details.sims.results.3.client.id": 60
"endpoints.0.details.sims.results.3.client.isReference": false
"endpoints.0.details.sims.results.3.client.name": "Android"
"endpoints.0.details.sims.results.3.client.version": "4.2.2"
"endpoints.0.details.sims.results.3.errorCode": 0
"endpoints.0.details.sims.results.3.protocolId": 769
"endpoints.0.details.sims.results.3.suiteId": 49170

"endpoints.0.details.sims.results.30.attempts": 1
"endpoints.0.details.sims.results.30.client.id": 27
"endpoints.0.details.sims.results.30.client.isReference": false
"endpoints.0.details.sims.results.30.client.name": "OpenSSL"
"endpoints.0.details.sims.results.30.client.version": "0.9.8y"
"endpoints.0.details.sims.results.30.errorCode": 0
"endpoints.0.details.sims.results.30.protocolId": 769
"endpoints.0.details.sims.results.30.suiteId": 22
"endpoints.0.details.sims.results.31.attempts": 1
"endpoints.0.details.sims.results.31.client.id": 99
"endpoints.0.details.sims.results.31.client.isReference": true
"endpoints.0.details.sims.results.31.client.name": "OpenSSL"
"endpoints.0.details.sims.results.31.client.version": "1.0.1l"
"endpoints.0.details.sims.results.31.errorCode": 0
"endpoints.0.details.sims.results.31.protocolId": 771
"endpoints.0.details.sims.results.31.suiteId": 49199
"endpoints.0.details.sims.results.32.attempts": 1
"endpoints.0.details.sims.results.32.client.id": 121
"endpoints.0.details.sims.results.32.client.isReference": true
"endpoints.0.details.sims.results.32.client.name": "OpenSSL"
"endpoints.0.details.sims.results.32.client.version": "1.0.2e"
"endpoints.0.details.sims.results.32.errorCode": 0
"endpoints.0.details.sims.results.32.protocolId": 771
"endpoints.0.details.sims.results.32.suiteId": 49199
"endpoints.0.details.sims.results.33.attempts": 1
"endpoints.0.details.sims.results.33.client.id": 32
"endpoints.0.details.sims.results.33.client.isReference": false
"endpoints.0.details.sims.results.33.client.name": "Safari"
"endpoints.0.details.sims.results.33.client.platform": "OS X 10.6.8"
"endpoints.0.details.sims.results.33.client.version": "5.1.9"
"endpoints.0.details.sims.results.33.errorCode": 0
"endpoints.0.details.sims.results.33.protocolId": 769
"endpoints.0.details.sims.results.33.suiteId": 49171
"endpoints.0.details.sims.results.34.attempts": 1
"endpoints.0.details.sims.results.34.client.id": 33
"endpoints.0.details.sims.results.34.client.isReference": true
"endpoints.0.details.sims.results.34.client.name": "Safari"
"endpoints.0.details.sims.results.34.client.platform": "iOS 6.0.1"
"endpoints.0.details.sims.results.34.client.version": "6"
"endpoints.0.details.sims.results.34.errorCode": 0
"endpoints.0.details.sims.results.34.protocolId": 771
"endpoints.0.details.sims.results.34.suiteId": 49191
"endpoints.0.details.sims.results.35.attempts": 1
"endpoints.0.details.sims.results.35.client.id": 34
"endpoints.0.details.sims.results.35.client.isReference": true
"endpoints.0.details.sims.results.35.client.name": "Safari"
"endpoints.0.details.sims.results.35.client.platform": "OS X 10.8.4"
"endpoints.0.details.sims.results.35.client.version": "6.0.4"
"endpoints.0.details.sims.results.35.errorCode": 0

"endpoints.0.details.sims.results.35.protocolId": 769
"endpoints.0.details.sims.results.35.suiteId": 49171
"endpoints.0.details.sims.results.36.attempts": 1
"endpoints.0.details.sims.results.36.client.id": 63
"endpoints.0.details.sims.results.36.client.isReference": true
"endpoints.0.details.sims.results.36.client.name": "Safari"
"endpoints.0.details.sims.results.36.client.platform": "iOS 7.1"
"endpoints.0.details.sims.results.36.client.version": "7"
"endpoints.0.details.sims.results.36.errorCode": 0
"endpoints.0.details.sims.results.36.protocolId": 771
"endpoints.0.details.sims.results.36.suiteId": 49191
"endpoints.0.details.sims.results.37.attempts": 1
"endpoints.0.details.sims.results.37.client.id": 35
"endpoints.0.details.sims.results.37.client.isReference": true
"endpoints.0.details.sims.results.37.client.name": "Safari"
"endpoints.0.details.sims.results.37.client.platform": "OS X 10.9"
"endpoints.0.details.sims.results.37.client.version": "7"
"endpoints.0.details.sims.results.37.errorCode": 0
"endpoints.0.details.sims.results.37.protocolId": 771
"endpoints.0.details.sims.results.37.suiteId": 49191
"endpoints.0.details.sims.results.38.attempts": 1
"endpoints.0.details.sims.results.38.client.id": 85
"endpoints.0.details.sims.results.38.client.isReference": true
"endpoints.0.details.sims.results.38.client.name": "Safari"
"endpoints.0.details.sims.results.38.client.platform": "iOS 8.4"
"endpoints.0.details.sims.results.38.client.version": "8"
"endpoints.0.details.sims.results.38.errorCode": 0
"endpoints.0.details.sims.results.38.protocolId": 771
"endpoints.0.details.sims.results.38.suiteId": 49191
"endpoints.0.details.sims.results.39.attempts": 1
"endpoints.0.details.sims.results.39.client.id": 87
"endpoints.0.details.sims.results.39.client.isReference": true
"endpoints.0.details.sims.results.39.client.name": "Safari"
"endpoints.0.details.sims.results.39.client.platform": "OS X 10.10"
"endpoints.0.details.sims.results.39.client.version": "8"
"endpoints.0.details.sims.results.39.errorCode": 0
"endpoints.0.details.sims.results.39.protocolId": 771
"endpoints.0.details.sims.results.39.suiteId": 49191
"endpoints.0.details.sims.results.4.attempts": 1
"endpoints.0.details.sims.results.4.client.id": 61
"endpoints.0.details.sims.results.4.client.isReference": false
"endpoints.0.details.sims.results.4.client.name": "Android"
"endpoints.0.details.sims.results.4.client.version": "4.3"
"endpoints.0.details.sims.results.4.errorCode": 0
"endpoints.0.details.sims.results.4.protocolId": 769
"endpoints.0.details.sims.results.4.suiteId": 49170
"endpoints.0.details.sims.results.40.attempts": 1
"endpoints.0.details.sims.results.40.client.id": 114
"endpoints.0.details.sims.results.40.client.isReference": true

"endpoints.0.details.sims.results.40.client.name": "Safari"
"endpoints.0.details.sims.results.40.client.platform": "iOS 9"
"endpoints.0.details.sims.results.40.client.version": "9"
"endpoints.0.details.sims.results.40.errorCode": 0
"endpoints.0.details.sims.results.40.protocolId": 771
"endpoints.0.details.sims.results.40.suiteId": 49199
"endpoints.0.details.sims.results.41.attempts": 1
"endpoints.0.details.sims.results.41.client.id": 111
"endpoints.0.details.sims.results.41.client.isReference": true
"endpoints.0.details.sims.results.41.client.name": "Safari"
"endpoints.0.details.sims.results.41.client.platform": "OS X 10.11"
"endpoints.0.details.sims.results.41.client.version": "9"
"endpoints.0.details.sims.results.41.errorCode": 0
"endpoints.0.details.sims.results.41.protocolId": 771
"endpoints.0.details.sims.results.41.suiteId": 49199
"endpoints.0.details.sims.results.42.attempts": 1
"endpoints.0.details.sims.results.42.client.id": 112
"endpoints.0.details.sims.results.42.client.isReference": true
"endpoints.0.details.sims.results.42.client.name": "Apple ATS"
"endpoints.0.details.sims.results.42.client.platform": "iOS 9"
"endpoints.0.details.sims.results.42.client.version": "9"
"endpoints.0.details.sims.results.42.errorCode": 0
"endpoints.0.details.sims.results.42.protocolId": 771
"endpoints.0.details.sims.results.42.suiteId": 49199
"endpoints.0.details.sims.results.43.attempts": 1
"endpoints.0.details.sims.results.43.client.id": 92
"endpoints.0.details.sims.results.43.client.isReference": false
"endpoints.0.details.sims.results.43.client.name": "Yahoo Slurp"
"endpoints.0.details.sims.results.43.client.version": "Jan 2015"
"endpoints.0.details.sims.results.43.errorCode": 0
"endpoints.0.details.sims.results.43.protocolId": 771
"endpoints.0.details.sims.results.43.suiteId": 49170
"endpoints.0.details.sims.results.44.attempts": 1
"endpoints.0.details.sims.results.44.client.id": 93
"endpoints.0.details.sims.results.44.client.isReference": false
"endpoints.0.details.sims.results.44.client.name": "YandexBot"
"endpoints.0.details.sims.results.44.client.version": "Jan 2015"
"endpoints.0.details.sims.results.44.errorCode": 0
"endpoints.0.details.sims.results.44.protocolId": 771
"endpoints.0.details.sims.results.44.suiteId": 49170
"endpoints.0.details.sims.results.5.attempts": 1
"endpoints.0.details.sims.results.5.client.id": 62
"endpoints.0.details.sims.results.5.client.isReference": false
"endpoints.0.details.sims.results.5.client.name": "Android"
"endpoints.0.details.sims.results.5.client.version": "4.4.2"
"endpoints.0.details.sims.results.5.errorCode": 0
"endpoints.0.details.sims.results.5.protocolId": 771
"endpoints.0.details.sims.results.5.suiteId": 49170
"endpoints.0.details.sims.results.6.attempts": 1

"endpoints.0.details.sims.results.6.client.id": 88
"endpoints.0.details.sims.results.6.client.isReference": false
"endpoints.0.details.sims.results.6.client.name": "Android"
"endpoints.0.details.sims.results.6.client.version": "5.0.0"
"endpoints.0.details.sims.results.6.errorCode": 0
"endpoints.0.details.sims.results.6.protocolId": 771
"endpoints.0.details.sims.results.6.suiteId": 49170
"endpoints.0.details.sims.results.7.attempts": 1
"endpoints.0.details.sims.results.7.client.id": 129
"endpoints.0.details.sims.results.7.client.isReference": false
"endpoints.0.details.sims.results.7.client.name": "Android"
"endpoints.0.details.sims.results.7.client.version": "6.0"
"endpoints.0.details.sims.results.7.errorCode": 0
"endpoints.0.details.sims.results.7.protocolId": 771
"endpoints.0.details.sims.results.7.suiteId": 49199
"endpoints.0.details.sims.results.8.attempts": 1
"endpoints.0.details.sims.results.8.client.id": 94
"endpoints.0.details.sims.results.8.client.isReference": false
"endpoints.0.details.sims.results.8.client.name": "Baidu"
"endpoints.0.details.sims.results.8.client.version": "Jan 2015"
"endpoints.0.details.sims.results.8.errorCode": 0
"endpoints.0.details.sims.results.8.protocolId": 769
"endpoints.0.details.sims.results.8.suiteId": 49169
"endpoints.0.details.sims.results.9.attempts": 1
"endpoints.0.details.sims.results.9.client.id": 91
"endpoints.0.details.sims.results.9.client.isReference": false
"endpoints.0.details.sims.results.9.client.name": "BingPreview"
"endpoints.0.details.sims.results.9.client.version": "Jan 2015"
"endpoints.0.details.sims.results.9.errorCode": 0
"endpoints.0.details.sims.results.9.protocolId": 771
"endpoints.0.details.sims.results.9.suiteId": 49170
"endpoints.0.details.sniRequired": false
"endpoints.0.details.stsPreload": false
"endpoints.0.details.stsResponseHeader": ""
"endpoints.0.details.stsStatus": "absent"
"endpoints.0.details.stsSubdomains": false
"endpoints.0.details.suites.list.0.cipherStrength": 128
"endpoints.0.details.suites.list.0.id": 4
"endpoints.0.details.suites.list.0.name": "TLS_RSA_WITH_RC4_128_MD5"
"endpoints.0.details.suites.list.1.cipherStrength": 128
"endpoints.0.details.suites.list.1.id": 5
"endpoints.0.details.suites.list.1.name": "TLS_RSA_WITH_RC4_128_SHA"
"endpoints.0.details.suites.list.10.cipherStrength": 128
"endpoints.0.details.suites.list.10.ecdhBits": 571
"endpoints.0.details.suites.list.10.ecdhStrength": 15360
"endpoints.0.details.suites.list.10.id": 49191
"endpoints.0.details.suites.list.10.name": "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256"
"endpoints.0.details.suites.list.11.cipherStrength": 128
"endpoints.0.details.suites.list.11.ecdhBits": 571

"endpoints.0.details.suites.list.11.ecdhStrength": 15360
"endpoints.0.details.suites.list.11.id": 49199
"endpoints.0.details.suites.list.11.name": "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256"
"endpoints.0.details.suites.list.12.cipherStrength": 168
"endpoints.0.details.suites.list.12.id": 10
"endpoints.0.details.suites.list.12.name": "TLS_RSA_WITH_3DES_EDE_CBC_SHA"
"endpoints.0.details.suites.list.13.cipherStrength": 168
"endpoints.0.details.suites.list.13.dhG": 128
"endpoints.0.details.suites.list.13.dhP": 128
"endpoints.0.details.suites.list.13.dhStrength": 1024
"endpoints.0.details.suites.list.13.dhYs": 128
"endpoints.0.details.suites.list.13.id": 22
"endpoints.0.details.suites.list.13.name": "TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA"
"endpoints.0.details.suites.list.14.cipherStrength": 168
"endpoints.0.details.suites.list.14.ecdhBits": 571
"endpoints.0.details.suites.list.14.ecdhStrength": 15360
"endpoints.0.details.suites.list.14.id": 49170
"endpoints.0.details.suites.list.14.name": "TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA"
"endpoints.0.details.suites.list.2.cipherStrength": 128
"endpoints.0.details.suites.list.2.id": 47
"endpoints.0.details.suites.list.2.name": "TLS_RSA_WITH_AES_128_CBC_SHA"
"endpoints.0.details.suites.list.3.cipherStrength": 128
"endpoints.0.details.suites.list.3.dhG": 128
"endpoints.0.details.suites.list.3.dhP": 128
"endpoints.0.details.suites.list.3.dhStrength": 1024
"endpoints.0.details.suites.list.3.dhYs": 128
"endpoints.0.details.suites.list.3.id": 51
"endpoints.0.details.suites.list.3.name": "TLS_DHE_RSA_WITH_AES_128_CBC_SHA"
"endpoints.0.details.suites.list.4.cipherStrength": 128
"endpoints.0.details.suites.list.4.ecdhBits": 571
"endpoints.0.details.suites.list.4.ecdhStrength": 15360
"endpoints.0.details.suites.list.4.id": 49169
"endpoints.0.details.suites.list.4.name": "TLS_ECDHE_RSA_WITH_RC4_128_SHA"
"endpoints.0.details.suites.list.5.cipherStrength": 128
"endpoints.0.details.suites.list.5.ecdhBits": 571
"endpoints.0.details.suites.list.5.ecdhStrength": 15360
"endpoints.0.details.suites.list.5.id": 49171
"endpoints.0.details.suites.list.5.name": "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA"
"endpoints.0.details.suites.list.6.cipherStrength": 128
"endpoints.0.details.suites.list.6.id": 60
"endpoints.0.details.suites.list.6.name": "TLS_RSA_WITH_AES_128_CBC_SHA256"
"endpoints.0.details.suites.list.7.cipherStrength": 128
"endpoints.0.details.suites.list.7.dhG": 128
"endpoints.0.details.suites.list.7.dhP": 128
"endpoints.0.details.suites.list.7.dhStrength": 1024
"endpoints.0.details.suites.list.7.dhYs": 128
"endpoints.0.details.suites.list.7.id": 103
"endpoints.0.details.suites.list.7.name": "TLS_DHE_RSA_WITH_AES_128_CBC_SHA256"
"endpoints.0.details.suites.list.8.cipherStrength": 128

"endpoints.0.details.suites.list.8.id": 156
"endpoints.0.details.suites.list.8.name": "TLS_RSA_WITH_AES_128_GCM_SHA256"
"endpoints.0.details.suites.list.9.cipherStrength": 128
"endpoints.0.details.suites.list.9.dhG": 128
"endpoints.0.details.suites.list.9.dhP": 128
"endpoints.0.details.suites.list.9.dhStrength": 1024
"endpoints.0.details.suites.list.9.dhYs": 128
"endpoints.0.details.suites.list.9.id": 158
"endpoints.0.details.suites.list.9.name": "TLS_DHE_RSA_WITH_AES_128_GCM_SHA256"
"endpoints.0.details.suites.preference": false
"endpoints.0.details.supportsNpn": false
"endpoints.0.details.supportsRc4": true
"endpoints.0.details.vulnBeast": true
"endpoints.0.duration": 115834
"endpoints.0.eta": 1
"endpoints.0.grade": "B"
"endpoints.0.gradeTrustIgnored": "B"
"endpoints.0.hasWarnings": true
"endpoints.0.ipAddress": "200.137.204.55"
"endpoints.0.isExceptional": false
"endpoints.0.progress": 100
"endpoints.0.serverName": "zimbra.ciar.ufg.br"
"endpoints.0.statusMessage": "Ready"
"engineVersion": "1.23.50"
"host": "zimbra.ciar.ufg.br"
"isPublic": false
"port": 443
"protocol": "HTTP"
"startTime": 1470370429172
"status": "READY"
"testTime": 1470370546721
]

A.2 Relatório dos 55 servidores TLS

Neste anexo é apresentado o relatório de inferência dos 55 servidores TLS na rede da Universidade Federal de Goiás. O relatório foi gerado pela ferramenta implementada para este trabalho (*ssl-lufg*), que analisa os arquivos *.json* gerados pela ferramenta *ssllabs-scan* e verifica os parâmetros de segurança utilizados: tamanho da chave privada, algoritmo de *hash*, validade do certificado e versões do protocolo. Baseado nestes parâmetros o servidor é avaliado e classificado como seguro ou inseguro e o porquê. No relatório há também informações referente ao nome de domínio presente no certificado digital, estas que não foram utilizadas para determinar a segurança dos servidores, mas nos permitiu identificar que alguns deles possivelmente sofreram ataques, pois possuem domínios diferentes aos pertencentes à UFG.

- Relatório da Análise de Segurança do TLS em Estações Servidoras na rede da Universidade Federal de Goiás -

```
****artemis.inf.ufg.br.json****
O tamanho da chave privada é de: 1024-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "www.inf.ufg.br"
Proprietário(subject):
"1.2.840.113549.1.9.1=#16127375706f72746540696e662e7566672e6272,CN=www.inf.ufg.br,O
U=Instituto
Emitente(issuer):
"1.2.840.113549.1.9.1=#16127375706f72746540696e662e7566672e6272,CN=www.inf.ufg.br,O
U=Instituto
Período de validade do certificado:
Fri Jan 09 10:29:27 BRST 2015 - Mon Jan 08 10:29:27 BRST 2018Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0
O servidor é inseguro porque a chave privada tem tamanho 1024-bits
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o certificado é auto assinado

        #==========#===========#==========#==========#===========#==========#

****ead.inf.ufg.br.json****
O tamanho da chave privada é de: 1024-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "ead.inf.ufg.br"
Proprietário(subject): "CN=ead.inf.ufg.br,OU=Instituto
Emitente(issuer): "CN=ead.inf.ufg.br,OU=Instituto
Período de validade do certificado:
Thu Aug 22 09:56:18 BRT 2013 - Sun Aug 21 09:56:18 BRT 2016Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque a chave privada tem tamanho 1024-bits
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o certificado é auto assinado

        #==========#===========#==========#==========#===========#==========#

****sgbd.inf.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "inf.ufg.br"
Proprietário(subject):
"1.2.840.113549.1.9.1=#16117375706f727440696e662e7566672e6272,CN=inf.ufg.br,OU=inf,
O=ufg,L=goiania,ST=go,C=br"
Emitente(issuer):
"1.2.840.113549.1.9.1=#16117375706f727440696e662e7566672e6272,CN=inf.ufg.br,OU=inf,
O=ufg,L=goiania,ST=go,C=br"
Período de validade do certificado:
Mon Apr 22 13:58:18 BRT 2013 - Tue Apr 22 13:58:18 BRT 2014Período de validade
excedido! Certificado inválido
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o certificado é auto assinado
```

```
        #==========#============#==========#==========#============#==========#


****portal.ufg.fibre.org.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ufg.fibre.org.br"
Proprietário(subject): "CN=*.ufg.fibre.org.br,O=REDE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Wed Sep 16 17:01:21 BRT 2015 - Sun Sep 16 17:01:21 BRT 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#============#==========#==========#============#==========#

****mail.labora.inf.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "labora.inf.ufg.br"
Proprietário(subject):
"1.2.840.113549.1.9.1=#161a736563757269747940 6c61626f72612e696e662e7566672e6272,CN=
labora.inf.ufg.br,OU=Labora,O=UFG,L=Goiania,ST=Goias,C=BR"
Emitente(issuer):
"1.2.840.113549.1.9.1=#161a736563757269747940 6c61626f72612e696e662e7566672e6272,CN=
labora.inf.ufg.br,OU=Labora,O=UFG,L=Goiania,ST=Goias,C=BR"
Período de validade do certificado:
Wed Jan 07 14:42:13 BRST 2015 - Sat Jan 04 14:42:13 BRST 2025Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o certificado é auto assinado

        #==========#============#==========#==========#============#==========#

****h20013720429.ufg.br.json****
O tamanho da chave privada é de: 1024-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "mail.funape.org.br"
Proprietário(subject): "CN=mail.funape.org.br,OU=Zimbra
Emitente(issuer): "CN=mail.funape.org.br,OU=Zimbra
Período de validade do certificado:
Tue Mar 25 14:49:36 BRT 2014 - Sun Mar 24 14:49:36 BRT 2019Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque a chave privada tem tamanho 1024-bits
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o certificado é auto assinado

        #==========#============#==========#==========#============#==========#

****marte.ciar.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ciar.ufg.br"
```

```
Proprietário(subject): "CN=*.ciar.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Mar 01 11:21:05 BRT 2016 - Sat Mar 02 11:21:05 BRT 2019Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****mercurio2.ciar.ufg.br.json****
O tamanho da chave privada é de: 1024-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "Mercurio-2.ciar.ufg.br"
Proprietário(subject): "CN=Mercurio-2.ciar.ufg.br"
Emitente(issuer): "CN=Mercurio-2.ciar.ufg.br"
Período de validade do certificado:
Tue Apr 27 08:22:07 BRT 2010 - Fri Apr 24 08:22:07 BRT 2020Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0
O servidor é inseguro porque a chave privada tem tamanho 1024-bits
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o certificado é auto assinado

        #==========#===========#==========#==========#===========#==========#

****h20013720570.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.face.ufg.br"
Proprietário(subject): "CN=*.face.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Sep 15 09:36:18 BRT 2015 - Sat Dec 15 10:36:18 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h20013721675.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "kett.galo.eco.br"
Proprietário(subject): "CN=kett.galo.eco.br"
Emitente(issuer): "CN=kett.galo.eco.br"
Período de validade do certificado:
Fri Nov 07 03:34:27 BRST 2014 - Sat Nov 07 03:54:27 BRST 2015Período de validade
excedido! Certificado inválido
Certificado é auto assinado!
Protocolos implementados: SSL 3.0 TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque o protocolo SSL 3.0 foi implementado.
O servidor é inseguro porque o certificado é auto assinado
```

```
        #==========#============#==========#==========#============#==========#

****h20013721677.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "DENNA.marceline.com"
Proprietário(subject): "CN=DENNA.marceline.com"
Emitente(issuer): "CN=DENNA.marceline.com"
Período de validade do certificado:
Tue Jul 21 15:55:39 BRT 2015 - Wed Jan 20 16:55:39 BRST 2016Período de validade
excedido! Certificado inválido
Certificado é auto assinado!
Protocolos implementados: SSL 3.0 TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o protocolo SSL 3.0 foi implementado.
O servidor é inseguro porque o certificado é auto assinado

        #==========#============#==========#==========#============#==========#

****shib.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ufg.br"
Proprietário(subject): "CN=*.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Aug 11 14:31:04 BRT 2015 - Sun Nov 11 15:31:04 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#============#==========#==========#============#==========#

****ns1.sectec.go.gov.br.json****
O tamanho da chave privada é de: 1024-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "NS1"
Proprietário(subject):
"1.2.840.113549.1.9.1=#161c696e666f726d617469636140736563746563 2e676f2e676f762e6272
,CN=NS1,OU=GERTIN,O=SECTEC,L=GOIANIA,ST=GOIAS,C=BR"
Emitente(issuer):
"1.2.840.113549.1.9.1=#161c696e666f726d617469636140736563746563 2e676f2e676f762e6272
,CN=NS1,OU=GERTIN,O=SECTEC,L=GOIANIA,ST=GOIAS,C=BR"
Período de validade do certificado:
Tue Sep 10 13:48:54 BRT 2013 - Wed Sep 10 13:48:54 BRT 2014Período de validade
excedido! Certificado inválido
Certificado é auto assinado!
Protocolos implementados:  TLS 1.2
O servidor é inseguro porque a chave privada tem tamanho 1024-bits
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o certificado é auto assinado

        #==========#============#==========#==========#============#==========#

****mail.ueg.edu.br.json****
O tamanho da chave privada é de: 1024-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "ueg.edu.br"
```

```
Proprietário(subject): "CN=ueg.edu.br,OU=Zimbra
Emitente(issuer): "CN=ueg.edu.br,OU=Zimbra
Período de validade do certificado:
Wed Sep 25 09:22:57 BRT 2013 - Mon Sep 24 09:22:57 BRT 2018Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados: SSL 3.0 TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque a chave privada tem tamanho 1024-bits
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o protocolo SSL 3.0 foi implementado.
O servidor é inseguro porque o certificado é auto assinado

        #==========#============#==========#==========#============#==========#

****horde5.inf.ufg.br.json****
O tamanho da chave privada é de: 1024-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "horde5.inf.ufg.br"
Proprietário(subject):
"1.2.840.113549.1.9.1=#16127375706f72746540696e662e7566672e6272,CN=horde5.inf.ufg.b
r,OU=Instituto
Emitente(issuer):
"1.2.840.113549.1.9.1=#16127375706f72746540696e662e7566672e6272,CN=horde5.inf.ufg.b
r,OU=Instituto
Período de validade do certificado:
Fri Jan 09 10:21:43 BRST 2015 - Mon Jan 08 10:21:43 BRST 2018Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0
O servidor é inseguro porque a chave privada tem tamanho 1024-bits
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o certificado é auto assinado

        #==========#============#==========#==========#============#==========#

****hades.inf.ufg.br.json****
O tamanho da chave privada é de: 1024-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "orion.inf.ufg.br"
Proprietário(subject):
"1.2.840.113549.1.9.1=#16147765626d617374657240696e662e7566672e6272,CN=orion.inf.uf
g.br,OU=INF,O=UFG,L=Goiania,ST=Goias,C=BR"
Emitente(issuer):
"1.2.840.113549.1.9.1=#16147765626d617374657240696e662e7566672e6272,CN=INF
Período de validade do certificado:
Mon Jun 07 22:44:23 BRT 2010 - Tue Jun 07 22:44:23 BRT 2011Período de validade
excedido! Certificado inválido
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0
O servidor é inseguro porque a chave privada tem tamanho 1024-bits
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1

        #==========#============#==========#==========#============#==========#

****dionisio.inf.ufg.br.json****
O tamanho da chave privada é de: 1024-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "sistemas.inf.ufg.br"
Proprietário(subject): "CN=sistemas.inf.ufg.br,OU=Instituto
```

```
Emitente(issuer): "CN=sistemas.inf.ufg.br,OU=Instituto
Período de validade do certificado:
Fri Apr 22 16:49:40 BRT 2016 - Mon Apr 20 16:49:40 BRT 2026Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque a chave privada tem tamanho 1024-bits
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o certificado é auto assinado

        #==========#===========#==========#==========#===========#==========#

****cia2.inf.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "cia2.inf.ufg.br"
Proprietário(subject): "CN=cia2.inf.ufg.br,OU=Instituto
Emitente(issuer): "CN=CIA2
Período de validade do certificado:
Wed Apr 09 09:33:45 BRT 2014 - Sat Apr 08 09:33:45 BRT 2017Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1

        #==========#===========#==========#==========#===========#==========#

****zabbix.ciar.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ciar.ufg.br"
Proprietário(subject): "CN=*.ciar.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Mar 01 11:21:05 BRT 2016 - Sat Mar 02 11:21:05 BRT 2019Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****zimbra.ciar.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ciar.ufg.br"
Proprietário(subject): "CN=*.ciar.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Mar 01 11:21:05 BRT 2016 - Sat Mar 02 11:21:05 BRT 2019Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h20013721678.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
```

```
O algoritmo de HASH utilizado é: SHA1
Common Names: "DENNA.marceline.com"
Proprietário(subject): "CN=DENNA.marceline.com"
Emitente(issuer): "CN=DENNA.marceline.com"
Período de validade do certificado:
Tue Jul 21 15:55:39 BRT 2015 - Wed Jan 20 16:55:39 BRST 2016Período de validade
excedido! Certificado inválido
Certificado é auto assinado!
Protocolos implementados: SSL 3.0 TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o protocolo SSL 3.0 foi implementado.
O servidor é inseguro porque o certificado é auto assinado

        #==========#===========#==========#==========#===========#==========#

****mail.cpa.evz.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "mail.cpa.evz.ufg.br"
Proprietário(subject): "CN=mail.cpa.evz.ufg.br,OU=Zimbra
Emitente(issuer): "CN=mail.cpa.evz.ufg.br,OU=Zimbra
Período de validade do certificado:
Thu May 22 11:16:09 BRT 2014 - Tue May 21 11:16:09 BRT 2019Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o certificado é auto assinado

        #==========#===========#==========#==========#===========#==========#

****h20013721748.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.sistemas.ufg.br"
Proprietário(subject): "CN=*.sistemas.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Sep 15 15:01:01 BRT 2015 - Sat Dec 15 16:01:01 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h20013721752.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.sistemas.ufg.br"
Proprietário(subject): "CN=*.sistemas.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Sep 15 15:01:01 BRT 2015 - Sat Dec 15 16:01:01 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!
```

```
        #==========#============#==========#==========#============#==========#

****h20013721754.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.sistemas.ufg.br"
Proprietário(subject): "CN=*.sistemas.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Sep 15 15:01:01 BRT 2015 - Sat Dec 15 16:01:01 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#============#==========#==========#============#==========#

****h200137217126.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "gi.fic.ufg.br"
Proprietário(subject): "CN=gi.fic.ufg.br"
Emitente(issuer): "CN=Let's
Período de validade do certificado:
Wed Apr 27 10:40:00 BRT 2016 - Tue Jul 26 10:40:00 BRT 2016Período de validade
excedido! Certificado inválido
Certificado não é auto assinado!
Protocolos implementados: SSL 3.0 TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque o protocolo SSL 3.0 foi implementado.

        #==========#============#==========#==========#============#==========#

****h200137217130.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.cercomp.ufg.br"
Proprietário(subject): "CN=*.cercomp.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Sep 15 08:26:05 BRT 2015 - Sat Dec 15 09:26:05 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#============#==========#==========#============#==========#

****h200137217132.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.cidarq.ufg.br"
Proprietário(subject): "CN=*.cidarq.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Sep 15 09:06:07 BRT 2015 - Sat Dec 15 10:06:07 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!
```

```
        #==========#===========#==========#==========#===========#==========#

****h200137217133.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ufg.br"
Proprietário(subject): "CN=*.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Aug 11 14:31:04 BRT 2015 - Sun Nov 11 15:31:04 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h200137217135.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.revistas.ufg.br"
Proprietário(subject): "CN=*.revistas.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Thu Dec 17 10:21:11 BRST 2015 - Mon Dec 17 10:21:11 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****mx.jatai.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.jatai.ufg.br"
Proprietário(subject): "CN=*.jatai.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Wed Nov 26 07:51:05 BRST 2014 - Fri Nov 27 07:51:05 BRST 2015Período de validade
excedido! Certificado inválido
Certificado não é auto assinado!
Protocolos implementados: SSL 3.0 TLS 1.0
O servidor é inseguro porque o protocolo SSL 3.0 foi implementado.
O servidor é inseguro porque o prazo de validade do certificado excedeu

        #==========#===========#==========#==========#===========#==========#

****h200137217156.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.bc.ufg.br"
Proprietário(subject): "CN=*.bc.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Sep 15 08:51:05 BRT 2015 - Sat Dec 15 09:51:05 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
```

```
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h200137217159.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "www.lapig.iesa.ufg.br"
Proprietário(subject): "CN=www.lapig.iesa.ufg.br,OU=EssentialSSL,OU=Domain
Emitente(issuer): "CN=COMODO
Período de validade do certificado:
Mon Jul 27 21:00:00 BRT 2015 - Wed Jul 27 20:59:59 BRT 2016Período de validade
excedido! Certificado inválido
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h200137217163.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "AH_EPOSERVER02"
Proprietário(subject): "CN=AH_EPOSERVER02,OU=ePO,O=McAfee"
Emitente(issuer): "CN=AH_CA_EPOSERVER02,OU=AH,O=McAfee"
Período de validade do certificado:
Wed Dec 31 21:00:00 BRT 1969 - Tue Feb 14 10:57:40 BRST 2045Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1

        #==========#===========#==========#==========#===========#==========#

****terra.eee.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "Gustavo"
Proprietário(subject):
"1.2.840.113549.1.9.1=#16166775737461766f6469617340656d632e7566672e6272,CN=Gustavo
Emitente(issuer):
"1.2.840.113549.1.9.1=#16166775737461766f6469617340656d632e7566672e6272,CN=Gustavo
Período de validade do certificado:
Mon Feb 01 10:45:52 BRST 2016 - Tue Jan 31 10:45:52 BRST 2017Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!
O servidor implementa 3 dos 4 requisitos de segurança TLS, mas é considerado
inseguro porque o certificado é auto assinado


        #==========#===========#==========#==========#===========#==========#

****redmine.cercomp.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.cercomp.ufg.br"
```

```
Proprietário(subject): "CN=*.cercomp.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Sep 15 08:26:05 BRT 2015 - Sat Dec 15 09:26:05 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h200137217196.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.cercomp.ufg.br"
Proprietário(subject): "CN=*.cercomp.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Sep 15 08:26:05 BRT 2015 - Sat Dec 15 09:26:05 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h200137217203.ufg.br.json****
O tamanho da chave privada é de: 1024-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "200.137.222.26"
Proprietário(subject): "CN=200.137.222.26,OU=UFG,O=CERCOMP,L=Goiania,ST=GO,C=BR"
Emitente(issuer):
"1.2.840.113549.1.9.1=#16176d61726369616e6f40636572636f6d702e7566672e6272,CN=Autori
dade
Período de validade do certificado:
Tue Nov 27 16:36:32 BRST 2012 - Wed Nov 27 16:36:32 BRST 2013Período de validade
excedido! Certificado inválido
Certificado não é auto assinado!
Protocolos implementados: SSL 3.0 TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque a chave privada tem tamanho 1024-bits
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o protocolo SSL 3.0 foi implementado.

        #==========#===========#==========#==========#===========#==========#

****h200137217205.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.cercomp.ufg.br"
Proprietário(subject): "CN=*.cercomp.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Sep 15 08:26:05 BRT 2015 - Sat Dec 15 09:26:05 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#
```

```
****oscercomp.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ufg.br"
Proprietário(subject): "CN=*.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Aug 11 14:31:04 BRT 2015 - Sun Nov 11 15:31:04 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h200137217219.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ufg.br"
Proprietário(subject): "CN=*.ufg.br,O=Universidade
Emitente(issuer): "CN=GlobalSign
Período de validade do certificado:
Mon May 12 10:47:02 BRT 2014 - Sun Nov 08 16:30:29 BRST 2015Período de validade
excedido! Certificado inválido
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor implementa 3 dos 4 requisitos de segurança TLS, mas é considerado
inseguro porque o período de validade do certificado excedeu

        #==========#===========#==========#==========#===========#==========#

****h200137218130.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ufg.br"
Proprietário(subject): "CN=*.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Aug 11 14:31:04 BRT 2015 - Sun Nov 11 15:31:04 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h200137218137.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.extras.ufg.br"
Proprietário(subject): "CN=*.extras.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Dec 01 10:26:02 BRST 2015 - Sat Dec 01 10:26:02 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!
```

```
        #==========#===========#==========#==========#===========#==========#

****h200137218139.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.weby.ufg.br"
Proprietário(subject): "CN=*.weby.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Wed Nov 11 15:01:02 BRST 2015 - Sun Nov 11 15:01:02 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****projetos.ufg.br.json****
O tamanho da chave privada é de: 1024-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "redmine-cegef"
Proprietário(subject): "CN=redmine-cegef"
Emitente(issuer): "CN=redmine-cegef"
Período de validade do certificado:
Thu Jun 24 15:02:59 BRT 2010 - Sun Jun 21 15:02:59 BRT 2020Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque a chave privada tem tamanho 1024-bits
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1
O servidor é inseguro porque o certificado é auto assinado

        #==========#===========#==========#==========#===========#==========#

****h200137218144.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ufg.br"
Proprietário(subject): "CN=*.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Aug 11 14:31:04 BRT 2015 - Sun Nov 11 15:31:04 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 3 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****mail.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ufg.br"
Proprietário(subject): "CN=*.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Aug 11 14:31:04 BRT 2015 - Sun Nov 11 15:31:04 BRST 2018Certificado dentro do
prazo de validade
```

```
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h200137218148.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.shibsp.ufg.br"
Proprietário(subject): "CN=*.shibsp.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Thu Aug 06 17:21:05 BRT 2015 - Mon Aug 06 17:21:05 BRT 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****ead.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ead.ufg.br"
Proprietário(subject): "CN=*.ead.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Mar 08 14:31:05 BRT 2016 - Sat Mar 09 14:31:05 BRT 2019Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****webmail.grad.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.grad.ufg.br"
Proprietário(subject): "CN=*.grad.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Mon Oct 26 14:16:04 BRST 2015 - Fri Oct 26 14:16:04 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****sistemas.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ufg.br"
Proprietário(subject): "CN=*.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Aug 11 14:31:04 BRT 2015 - Sun Nov 11 15:31:04 BRST 2018Certificado dentro do
```

```
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h20013722185.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.ufg.br"
Proprietário(subject): "CN=*.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Aug 11 14:31:04 BRT 2015 - Sun Nov 11 15:31:04 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h200137221168.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA1
Common Names: "*.ufg.br"
Proprietário(subject): "CN=*.ufg.br,O=Universidade
Emitente(issuer): "CN=GlobalSign
Período de validade do certificado:
Thu Nov 07 16:30:29 BRST 2013 - Sun Nov 08 16:30:29 BRST 2015Período de validade
excedido! Certificado inválido
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é inseguro porque o algoritmo de hash utilizado é o SHA1

        #==========#===========#==========#==========#===========#==========#

****h200137221180.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "*.extras.ufg.br"
Proprietário(subject): "CN=*.extras.ufg.br,O=UNIVERSIDADE
Emitente(issuer): "CN=ICPEdu,O=Rede
Período de validade do certificado:
Tue Dec 01 10:26:02 BRST 2015 - Sat Dec 01 10:26:02 BRST 2018Certificado dentro do
prazo de validade
Certificado não é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor é seguro porque implementa os 4 requisitos de segurança TLS!

        #==========#===========#==========#==========#===========#==========#

****h200137222222.ufg.br.json****
O tamanho da chave privada é de: 2048-bits
O algoritmo de HASH utilizado é: SHA256
Common Names: "Web"
Proprietário(subject): "CN=Web,OU=Cercomp,O=UFG,L=Goiania,ST=Goias,C=BR"
Emitente(issuer): "CN=Web,OU=Cercomp,O=UFG,L=Goiania,ST=Goias,C=BR"
Período de validade do certificado:
```

```
Thu Aug 07 14:46:46 BRT 2014 - Sun Aug 04 14:46:46 BRT 2024Certificado dentro do
prazo de validade
Certificado é auto assinado!
Protocolos implementados:  TLS 1.0 TLS 1.1 TLS 1.2
O servidor implementa 3 dos 4 requisitos de segurança TLS, mas é considerado
inseguro porque o certificado é auto assinado

        #==========#===========#==========#==========#===========#==========#
```