COMNET 4nd Edition Volume 1

Authentication and Authorization for Constrained Environments (ACE) com Framework OAuth e Protocolo CoAP

Cleber B. da Porciúncula[1], Sílvio Beskow[1], Érico Santos Rocha[1], Jeferson Campos Nobre[1]

[1]Universidade do Vale do Rio dos Sinos (Unisinos)
Porto Alegre – RS – Brasil

{cleber.bp,ericomsr.professor,segbeskow}@gmail.com, jcnobre@unisinos.br

Abstract. *The purpose of this article is to study the Authentication and Authorization for Constrained Environments (ACE) framework using the OAuth 2.0 framework and the CoAP protocol. The analyzed group explores concepts such as authentication and authorization in connection-restricted environments. This framework, encompasses a set of protocols that are under study for a standardization in the use in IoT devices. Restricted devices form communication networks exchanging information between each other, requiring security requirements to ensure confidentiality, integrity and availability.*

Resumo. *O presente artigo tem por objetivo estudar o framework ACE (Authentication and Authorization for Constrained Environments) utilizando o framework OAuth 2.0 e o protocolo CoAP. O grupo analisado explora conceitos como autenticação e autorização em ambientes restritos a conexão. Este framework, engloba um conjunto de protocolos que estão em estudo para uma padronização na utilização em dispositivos IoT. Dispositivos restritos formam redes de comunicação trocando informações entre si, necessitando de requisitos de segurança para a garantir confidencialidade, integridade e disponibilidade.*

1. Introdução

O IETF (*The Internet Engineering Task Force*) é um grupo informal e organizado que contribui para a evolução da engenharia de redes, além de desenvolver e aprimorar especificações, padrões, e tecnologias associadas a Internet. Através de *Working Groups-WG* (Grupos de Trabalho), novos padrões e tecnologias são consideradas, analisadas, e, por consenso, a proposta poderá se tornar um novo padrão de tecnologia oferecido pelo IETF [Hoffman 2013].

O WG ACE (*Working Group Authentication and Authorization for Constrained Environments*), propõe o uso de autenticação e autorização (com base no OAuth2.0) em dispositivos restritos, que formam os ambientes restritos [Kaduk et al. 2014]. Recomenda a utilização do protocolo CoAP (*Constrained Application Protocol* - [Shelby et al. 2014]) para a transmissão e o protocolo DTLS para a proteção (*Datagram Transport Layer Security*) [Rescorla and Modadugu 2012]. Além de garantir esses requisitos de segurança, pretende não afetar o tempo de resposta destes dispositivos, tornando a experiência do usuário agradável.

O artigo está organizado da seguinte forma: sessão 2 Autenticação e Autorização para Ambientes Restritos (ACE); sessão 3, Fundamentação Teórica; sessão 4, Conclusão e Trabalhos Futuros.

2. Autenticação e Autorização para Ambientes Restritos (ACE)

Ambientes restritos computacionalmente fornecem um desafio maior para a execução de aplicações e protocolos, pois precisam processar informações com garantia de qualidade na entrega de dados.

O ACE é um *framework* de um WG do IETF baseado nas interações do protocolo OAuth 2.0, usando *tokens* e terminais que não interagem com o ambiente. O objetivo é fornecer o uso padronizado dos requisitos de segurança, autenticação e autorização através dos protocolos OAuth 2.0 e CoAP em ambientes aonde não é possível utilizar o HTTP.

A estrutura garante que os dispositivos com restrições de recursos, que conectados entre si compõem as redes de componentes restritos, possam ser autenticados e autorizados, respectivamente, para acessar determinados recursos do sistema.

A especificação definida no *framework* ACE define requisitos de autorização para o ambiente IoT. O Framework ACE é formado por 4 blocos básicos: OAuth 2.0, CoAP, CBOR e COSE.

No OAuth 2.0, a comunicação entre o *token*, o cliente e o *authentication service* (AS) é realizada via uma URI. O *framework* ACE **recomenda** o uso do CoAP e uma alternativa para o uso da URI: o cliente utiliza uma mensagem CBOR enviando o *payload* via requisição *POST*.

O OAuth 2.0 utiliza mensagem JSON [T. Bray 2014] no *payload* como resposta para o cliente; o ACE **exige** o uso do CBOR e, dependendo do conteúdo, utiliza algum tipo de criptografia.

O *framework* é composto por perfis e extensões: o ***Credential Provisioning*** assume a falta de uma infraestrutura comum de distribuição de chaves. São providas credenciais de autenticação mútua pela AS.

Um ***Proof-of-Possession*** por padrão, implementa a prova de posse, vinculando uma chave ao *token*, que comprova sua posse por um dispositivo; ***ACE Profiles*** fazem limitação dos códigos ou protocolos que um *Client* (C) ou *Resource Server* (RS) suporta.

Um ***Authentication Server*** (AS) gerencia e compara a compatibilidade das escolhas feitas por um determinado cliente durante a sua interação com um RS. O AS especifica por parâmetros no *token* de resposta, como as autenticações mútuas são feitas entre C e o RS; ***OAuth 2.0*** é NECESSÁRIO para que os dados trocados com o AS sejam criptografados e protegidos por integridade.

Além disso, é NECESSÁRIO que o AS e o ponto final se comuniquem com ele (cliente ou RS) e realizem a autenticação mútua. RECOMENDA-SE o uso do CoAP como alternativa aos parâmetros URI: o remetente (cliente ou RS) codifica os parâmetros de sua solicitação como um mapa CBOR e envia esse mapa como a carga útil do pedido POST [Kaduk et al. 2014].

Utiliza os modos de operação a seguir: **Descobrindo servidores de Autenticação**, para determinar o AS encarregado de um recurso hospedado no RS, C pode enviar uma mensagem inicial de Solicitação de Recursos Não Autorizados para RS.

Deste modo, o RS então nega o pedido e envia o endereço do AS de volta para C; uma **Concessão de Autorização**, para solicitar um token de acesso, o cliente obtém au-

torização do proprietário do recurso ou usa suas credenciais de clientes como concessão.

A autorização é expressa sob a forma de uma concessão de autorização; um **Token de Introspecção (Opcional)** é uma função opcional e fornecida pelo AS. O token é usado pelo RS ou cliente para consultar o AS para metadados sobre um determinado token, como, por exemplo, validade ou escopo. Adaptando para ambientes restritos, usa-se o CBOR.

Um **Token de Acesso** neste *framework* recomenda o uso do CBOR web *token* (CWT) como especificado em [ID.ietf-ace-cbor-web-token]. O objetivo é de facilitar o processamento *offline* do acesso dos *tokens*, requisita-se o "cnf"de [I-D.ietf-ace-cwt-proof-of-possession] e especifica a requisição "*scope*"para os *tokens* da Web JSON e CBOR, isso permite acesso aos *tokens* [Kaduk et al. 2014].

A Figura 1 ilustra este processo de autenticação e autorização a recursos solicitados por um cliente na arquitetura ACE.

A utilização do CoAP em conjunto com o DTLS propõe uma alternativa viável para a solução de autenticação e autorização neste tipo de dispositivo restrito. O CoAP pode ser utilizado de 4 maneiras:

1. **NoSec**: sem segurança, DTLS desabilitado, uso de IPSec recomendável
2. **PreSharedKey**: DTLS habilitado, utiliza chaves compartilhadas simétricas, conforme RFC4279 [P. Eronen and H. Tschofenig 2005]
3. **RawPublicKey**: DTLS habilitado, utiliza chaves assimétricas sem um certificado digital, conforme RFC7250 [P. Wouters et al. 2014]
4. **Certificate**: DTLS habilitado utilizando certificado digital X.509 conforme RFC5280 [Cooper et al. 2008]

O CoAP é um protocolo de serviço em que as mensagens são encapsuladas e transmitidas através do *payload* do UDP. O CoAP utiliza a porta 5683 como porta padrão.

Como o HTTP, o CoAP utiliza o *framework* OAuth 2.0 para realizar o processo de autenticação, fazendo o uso de *tokens* para conceder o privilégio necessário ao recurso solicitado.

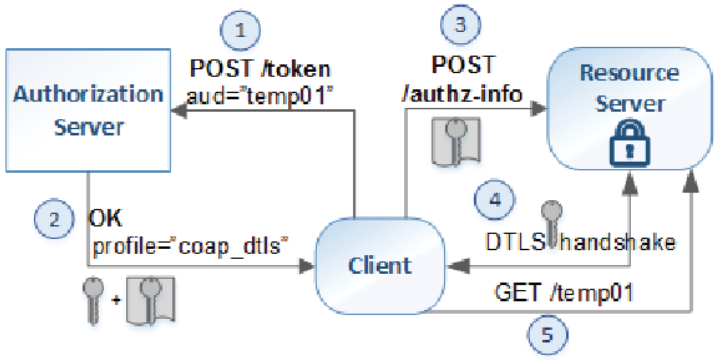
Já o DTLS é utilizado para a comunicação segura dos dados entre dispositivos, pois preserva as características do UDP, fornecendo o requisito de autorização, garantido segurança na troca de mensagens entre dispositivos.

A Figura 1 ilustra o funcionamento do *framework* ACE em conjunto com os protocolos DTLS e CoAP.

O Cliente solicita uma autenticação ao AS (*Authorization Server*) antes de solicitar acesso a um recurso do RS (*Resource Server*); se a autenticação for concedida, o AS fornece ao cliente um *token* de acesso.

De posse do *token* de acesso, o Cliente poderá solicitar acesso a um recurso, apresentando este *token* ao RS. O RS não conhece o cliente mas confia no AS e portanto passa a confiar no cliente indiretamente, pois este recebeu um *token* de acesso do AS.

Em uma arquitetura IoT tanto o cliente como o RS trabalham de forma independente na maioria das vezes, sem intervenção humana.



:. 2. Outline of delegated authorization with ACE configured with transpo

Figura 1. ACE, OAuth 2.0, DTLS - Fonte: [Beltran and Gómez-Skarmeta 2016]

A segurança desta comunicação depende da utilização de um protocolo criptográfico como o DTLS para garantir a troca de informações de maneira segura.

3. Fundamentação Teórica

3.1. Dispositivos Restritos

Um dispositivo restrito computacionalmente possui restrições de energia, de armazenamento, de capacidade de transmissão e de processamento. Estes dispositivos são projetados, em sua maior parte, sem uma interface de comunicação com o usuário, destinados a realizar uma interação sem a utilização da intervenção humana [Seitz et al. 2016].

Estes dispositivos podem formar uma rede, e esta rede pode apresentar características restritas, como canal não confiável, com perdas, limitação de largura de banda e uma topologia altamente dinâmica [Bormann et al. 2014]. Em ambientes restritos, os requisitos de autenticação e autorização representam desafios para a utilização de protocolos.

Os dispositivos restritos podem ser classificados conforme a Tabela 1:

Nome	Data Size	Code Size	Característica
Class 0, C0	«10KiB	«100KiB	restrição severa, usados como sensores
Class 1, C1	∼10KiB	∼100KiB	restrição alta, protocolos específicos (CoAP)
Class 2, C2	∼50KiB	∼250KiB	restrição baixa, pode-se utilizar mais de um protocolo

Tabela 1. Classificação não definitiva de Dispositivos Restritos adaptado da RFC7228 - Fonte: RFC7228 [Bormann et al. 2014]

Os dispositivos ***Class 0*** são basicamente sensores, pois seus recursos de processamento e memória são extremamente limitados, não possuindo capacidade de comunicação com a Internet de forma segura. É necessário a utilização de dispositivos com maiores recursos computacionais como *proxies*, *gateways* ou servidores, que permitirão que os dispositivos ***Class 0*** possam realizar a sua comunicação com o meio externo. O gerenciamento destes dispositivos não é realizado de maneira tradicional; os dispositivos são pré-configurados com um conjunto de informações básicas (e raramente reconfigurados) e sua iteração com o meio externo é realizada através de sinais simples em resposta a uma solicitação de dados.

Os dispositivos ***Class 1*** possuem restrições de processamento e memória, o que tornam a sua comunicação com a Internet bastante difícil, quando necessitam utilizar uma pilha de protocolos completa como o HTTP [R. Fielding and J. Reschke 2014], TLS [Dierks and Rescorla 2008] e XML. Porém, os dispositivos ***Class 1*** podem utilizar uma pilha de protocolos específica para ambientes restritos, como o CoAP sobre UDP. Os dispositivos podem oferecer funções de suporte à segurança necessárias para a utilização em conjunto com uma grande rede, podendo integrar-se a uma rede IP, mas a utilização de seus recursos deve ser realizada de forma econômica, devido as suas restrições de memória e processamento.

Os dispositivos ***Class 2*** são menos limitados em recursos e são capazes de suportar a maioria das pilhas de protocolos utilizadas em computadores e servidores. Contudo, os dispositivos ***Class 2*** podem se beneficiar da utilização de protocolos para ambientes restritos, pois tais protocolos possuem uma maior eficiência energética, além de consumir menor largura de banda. Utilizando menos recursos, mais recursos tornam-se disponíveis para aplicações, reduzindo custos de desenvolvimento e aumentando a sua interoperabilidade.

Tais dispositivos com limitações de recursos como CPU, memória, energia e comprimento de banda, podem formar uma rede de componentes restritos, utilizados como sensores ou dispositivos inteligentes. Tais dispositivos em rede, podem possuir restrições como perda de canais, limitações de banda e possuir uma topologia altamente dinâmica, com a inclusão ou a saída de dispositivos. Os dispositivos, realizam coleta de informações em diversos ambientes como prédios, fábricas, florestas, hospitais, submersos e outros, trabalhando sob inúmeras restrições como energia, memória, consumo de banda e habilidade de comunicação. A RFC7102 [Vasseur 2014] utiliza o termo LLN (*Low-Power and Lossy Network*) para descrever as várias restrições em uma rede de dispositivos restritos.

3.2. OAuth 2.0

É um *framework* [D. Hardt 2012] de base para o ACE que suporta o requisito de segurança de autorização utilizando o protocolo TLS [Dierks and Rescorla 2008] que garante a confidencialidade dos dados que trafegam na Web. O *framework* define quatro papéis durante o seu uso:

1. Proprietário do recurso (*resource owner*)
2. Servidor de recurso (*resource server*)
3. Cliente (*client*)
4. Servidor de autorização (*authorization server*)

O *framework* introduz uma camada de autorização entre o cliente e o proprietário do recurso. O cliente faz uma solicitação de recursos controlados pelo proprietário do recurso onde estes recursos estão hospedados em um servidor de recursos. Para acessar os recursos desejados, o cliente emite um conjunto de credenciais diferentes daqueles do proprietário do recurso. Desta maneira, o cliente não utiliza as credenciais do proprietário do recurso para acessar o recurso requisitado; o cliente recebe um *token* de acesso, um componente que possui características específicas para acesso ao recurso requisitado, como tempo de vida, nível de privilégio, tipo de acesso entre outras características.

Os *tokens* de acesso são emitidos para o cliente através de um servidor de autorização com a aprovação do proprietário do recurso. O cliente utiliza o *token* obtido

para acessar o recurso solicitado hospedado pelo servidor de recursos. A especificação do *framework* é para a utilização somente com o protocolo HTTP [Fielding et al. 1999].

A Figura 2 ilustra os componentes e o fluxo de troca de informações no framework OAuth 2.0.

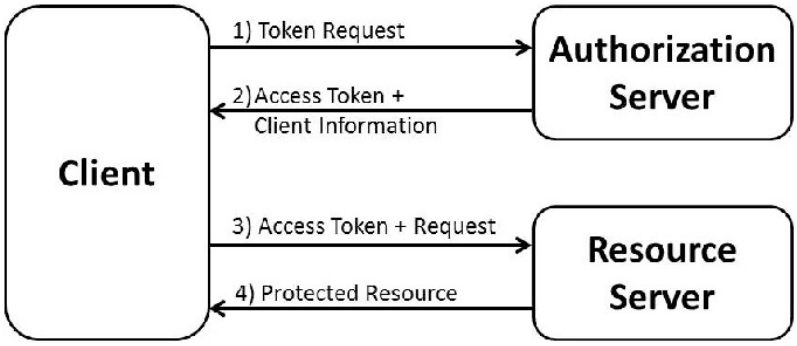


Figura 2. Mensagens e Componentes OAuth2.0 - Fonte: [Navas et al. 2016].

3.3. Constrained Application Protocol(CoAP)

O protocolo CoAP, definido na RFC7252 [Shelby et al. 2014] é um protocolo de transferência de informações projetado para o ambiente Web, para utilização em ambientes com atributos restritos como um ambiente sujeito a perdas, baixa potência de transmissão e processamento. Esse protocolo suporta dispositivos com pouca memória RAM/ROM, altas taxas de erro e é usado em redes pessoais e de baixa taxa de transferência. Além disso, é projetado para utilização em redes M2M (*machine-to-machine*), com baixo consumo de energia e alto grau de automação.

O CoAP integra-se com o protocolo HTTP [R. Fielding and J. Reschke 2014], com suporte a *multicast*, baixa sobrecarga e fácil utilização em ambientes restritos. Características: satisfaz os requisitos para protocolos Web M2M, possui protocolo UDP [Postel 1980] obrigatório com suporte a multicast e com unicast opcional permite troca de mensagens assíncrona, tem cabeçalho simples e de baixa complexidade, permite HTTP *stateless*, possibilitando que *proxies* acessem recursos CoAP via HTTP, dispõe de conexão segura com o uso de DTLS.

No CoAP a troca de mensagens é similar ao HTTP. Em interações M2M o protocolo deve estar ativo em ambos os lados, cliente/servidor. A requisição para troca de mensagem é semelhante a uma requisição HTTP, enviada pelo cliente requisitando uma ação (*Method Code*) ou recurso (uma URI) para o servidor. O servidor envia uma resposta (*Response Code*) onde esta resposta, pode incluir a representação de um recurso.

O CoAP utiliza troca de mensagens assíncronas por meio do UDP. Essa troca de mensagem é realizada logicamente através de uma camada de transporte que utiliza confiabilidade opcional. São definidos 4 tipos de mensagens: *Confirmable, Non-confirmable, Acknowledgement, Reset*.

O protocolo CoAP é entendido através de uma abordagem de 2 camadas: a que interage com o protocolo UDP e de requisição/resposta com a camada de aplicação, conforme definido na RFC7252 [Shelby et al. 2014].

Tanto a ação (*Method Code*) como a resposta (*Response Code*) podem transportar requisições e respostas. Na interação dos 4 tipos de mensagens, requisições podem transportar mensagens de confirmação e não-confirmação, bem como mensagens de resposta podem transportar mensagens de conexão.

A troca de mensagens no CoAP utiliza pacotes compactos, onde as mensagens ocupam a seção de dados do datagrama UDP. Por possuir um pacote de tamanho reduzido, o CoAP pode ser transmitido por outros protocolos: SMS, TCP, SCTP. Suas mensagens possuem um formato simples, conforme mostra a Figura 3.

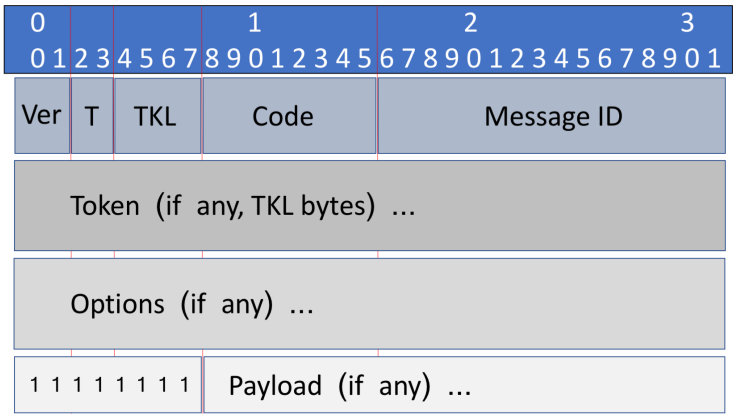


Figura 3. Formato da Mensagem CoAP - Fonte: RFC7252 [Shelby et al. 2014].

O significado dos campos do cabeçalho do protocolo é descrito a seguir:

Ver (*version*) - número da versão do protocolo

T (*type*) - tipo da mensagem, *Confirmable* (0), *Non-confirmable* (1), *Acknowledgement (2)*, ou *Reset* (3)

TKL (*token length*) - tamanho do *token*

Code - código de requisição/resposta

MessageID - detecção dos tipos de mensagem do protocolo

Token - valor para correlacionar requisições/respostas

Options - informações opcionais de controle do protocolo

Payload - carga útil a ser transmitida

3.4. Datagram Transporte Layer Security (DTLS)

O protocolo CoAP não possui mecanismo de segurança próprio, contudo seu uso foi pensado para garantir os requisitos de Autenticação e Autorização utilizando o *framework* OAuth 2.0.

Ele é utilizado em conjunto com os protocolos DTLS [Rescorla and Modadugu 2012] e IPSec [Bormann 2012] (opcional) para garantir a confidencialidade das informações.

O DTLS oferece as características de privacidade para o transporte de datagramas e permite que a troca de mensagem entre cliente/servidor estejam protegidas contra espionagem, adulteração e falsificação de mensagens.

É baseado no TLS (Transport Layer Security), RFC5246 [Dierks and Rescorla 2008] e possui o objetivo básico de utilizar o protocolo TLS em conjunto com o UDP, pois devido a característica do UDP, onde pacotes podem ser perdidos ou reorganizados, e o TLS não possuir mecanismos para suportar esta falta de confiabilidade.

Um número crescente de protocolos da camada de aplicação está sendo projetado para usar o UDP. Em particular, protocolos como SIP [Rosenberg et al. 2012] e protocolos de jogos eletrônicos, que são cada vez mais populares.

A filosofia básica de projeto do DTLS é utilizar o TLS sobre datagramas UDP. A razão pela qual o TLS não pode ser usado diretamente com datagramas UDP é devido aos pacotes serem perdidos ou reordenados.

O protocolo TLS não foi projetado para tratar este tipo de insegurança, portanto as implementações do protocolo não suportam as características do protocolo UDP.

A falta de confiabilidade do UDP cria dois tipos de problemas para o TLS: o TLS não permite a verificação independente de datagramas; o TLS assume que as mensagens de *handshake* são entregues de maneira confiável

Para resolver estes problemas, o DTLS utiliza um tempo para gerenciar a perda de pacotes; se uma mensagem de confirmação for perdida, o cliente saberá que esta será retransmitida.

No DTLS, cada mensagem de *handshake* possui um número de sequência, permitindo que esta mensagem seja processada se for a sequencia correta, ou será armazenada em uma fila, até que todas as mensagens anteriores tenham sido recebidas.

3.5. Concise Binary Object Representation(CBOR)

O CBOR é um formato de dados cujo objetivo é ter um código e uma mensagem muito pequenos, permitindo extensões sem a necessidade de negociar a versão [Bormann and Hoffman 2013]. O código deve ser simples, facilmente codificado/decodificado, sem a necessidade de esquemas, e deve suportar sistemas com limitações de recursos.

Em conjunto com o CBOR, pode ser utilizado o formato de dados encriptado, COSE - CBOR Object Signing and Encryption, uma especificação que define como criar/processar assinaturas, autenticar mensagens e utilizar criptografia usando o formado CBOR [Schaad and Cellars 2017]. Ele pode descrever a utilização de chaves de criptografia utilizando o CBOR

3.6. Segurança da Informação

A Segurança da Informação (SI) está alicerçada na tríade confidencialidade, integridade e disponibilidade. Ela se aprofunda e se enraíza em todas as áreas que tenham a informação como ativo a ser protegido. A SI não está restrita somente a CID. No artigo de [Simmonds et al. 2004], existe dois mnemônicos que são usados para resumir serviços

que devem ser providos por ela: a) o clássico CID e b) o triplo A – *Authentication, Authorization e Accounting* (Autenticação, Autorização e Conta). Uma pessoa não pode usar os serviços de uma conta de usuário até ter sido autorizado e previamente autenticado.

No WG *ACE*, apenas dois requisitos de segurança estão sendo abordados: a Autenticação e a Autorização. Esses requisitos são comentados a seguir.

1. **Autenticação** é um mecanismo que garante que uma entidade (pessoa, organização, aplicação) é quem afirma ser: a) **Autenticação de entidade par** - confirma a identidade de uma entidade. Sua finalidade é evitar que uma entidade ilegítima se disfarce ou repita dados de forma não autorizada; b) **Autenticação da origem de dados** - confirma a legitimidade da origem dos dados. Porém, não garante que esses dados não tenham sido manipulados antes.
2. **Autorização** é um mecanismo que após a entidade ser autenticada, seus privilégios com relação aos recursos (Ler, Gravar, Modificar ou Controle total), dentro do sistema, são definidos em uma base centralizada através da autorização.

Segundo Stallings [Stallings 2014], o controle de acesso fornece ao sistema uma forma de controlar e limitar os acessos de uma determinada entidade aos sistemas e aplicações.

A Tabela 2 mostra uma visão geral dos ambientes e protocolos para autenticação e autorização. Nesta tabela é realizada uma comparação entre a pilha de protocolos da Internet padrão com a pilha de protocolos proposta para a IoT.

Camadas	Ambiente Restrito OSI	Internet OSI	Ambiente Restrito Protocolo SEG	Internet Protocolo SEG
Aplicação	CoAP Requests/Responses Messages	HTTP	CoAPs Requests/Responses Messages	HTTPs
Transporte	UDP	TCP	DTLS	TLS
Rede	IPv6/6LowPAN	IP	IP/Ipsec/HIP	IP/Ipsec/HIP
Link	MAC	MAC	MAC	MAC
Física	PHY	PHY	PHY	PHY

Tabela 2. Comparação dos protocolos de comunicação e de segurança entre a Internet e a IoT e as camadas OSI clássicas - Fonte: Editada e traduzida pelo autor do original [Cirani et al. 2013]

Na camada de aplicação para a IoT, o protocolo CoAP/CoAPs é utilizado no lugar do HTTP/HTTPS; o protocolo UDP/DTLS é utilizado no lugar do TCP/TLS e na camada de rede, o protocolo IPv6/6LowPAN substitui o protocolo IP/IPsec/HIP.

A utilização dos protocolos citados na arquitetura da IoT no lugar dos protocolos tradicionais, pode ser sugerida devido as otimizações e adaptações necessárias a troca de informações entre dispositivos que possuem poucos recursos de processamento.

As versões dos protocolos adaptadas para utilização em dispositivos restritos podem utilizar um MTU (*Maximum Transmission Unit*) pequeno, reduzindo a fragmentação de pacotes diminuindo o atraso em transmissões. Pacotes menores consomem menores recursos, o que pode evitar um consumo alto para dispositivos alimentados por bateria.

Versões minimizadas e otimizadas de protocolos para a IoT, utilizados em todas as camadas, podem reduzir número de mensagens trocadas entre dispositivos, contribuindo no consumo de recursos e na segurança da troca de informações.

4. Conclusão e Trabalhos Futuros

O Grupo de Trabalho ACE propõe a padronização do protocolo CoAP como elemento que deve ser utilizado na troca de informações entre dispositivos M2M e IoT. É essencial a necessidade de uma melhor classificação de dispositivo restrito (itens como energia, ambiente, banda, eletromagnetismo, função, tamanho da palavra 8/16/32 bits, devem ser considerados) para um melhor aproveitamento das características do *framework*.

O protocolo CoAP possui bibliotecas [1] e implementações em linguagens de programação [2], facilitando o aprendizado e popularizando a utilização do protocolo.

Ao utilizar CoAP com o DTLS, a conexão é mantida aberta, pois abrir/fechar a conexão nesses protocolos torna a troca de mensagens ineficiente. Um estudo mais aprofundado sobre quais são os impactos relativos à Segurança da Informação pode contribuir para o aperfeiçoamento do *framework*.

O WG ACE ao propor o protocolo CoAP (baseado no HTTP) para troca de informações entre dispositivos IoT e o protocolo DTLS (baseado no TLS), para a segurança destas informações, permite que a curva de aprendizado destes protocolos seja menor, facilitando a sua implementação e utilização.

Como trabalho futuro, a implementação dos protocolos CoAP e DTLS em dispositivos IoT reais, como os microprocessadores Arduino e ESP8266, poderá contribuir para uma análise de desempenho.

Em um ambiente com dispositivos restritos reais, a realização de testes de segurança também poderá contribuir para identificar e solucionar eventuais falhas dos protocolos.

Referências

[Beltran and Gómez-Skarmeta 2016] Beltran, V. and Gómez-Skarmeta, A. F. (2016). An overview on delegated authorization for coap: Authentication and authorization for constrained environments (ace). *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 706–710.

[Bormann 2012] Bormann, C. (2012). Using CoAP with IPsec draft-bormann-core-ipsec-for-coap-00. Draft 00, CoRE Working Group.

[Bormann et al. 2014] Bormann, C., Ersue, M., and Keranen, A. (2014). Terminology for Constrained-Node Networks. RFC 7228, RFC Editor.

[Bormann and Hoffman 2013] Bormann, C. and Hoffman, P. (2013). Concise Binary Object Representation (CBOR). RFC 7049, RFC Editor.

[Cirani et al. 2013] Cirani, S., Ferrari, G., and Veltri, L. (2013). Enforcing security mechanisms in the ip-based internet of things: An algorithmic overview. *Algorithms*, 6(2):197–226.

[1] `https://libcoap.net/`

[2] `http://coap.technology/impls.html`

[Cooper et al. 2008] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and Polk, W. (2008). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, RFC Editor.

[D. Hardt 2012] D. Hardt, E. (2012). The OAuth 2.0 Authorization Framework. RFC 6749, RFC Editor.

[Dierks and Rescorla 2008] Dierks, T. and Rescorla, E. (2008). The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, RFC Editor.

[Fielding et al. 1999] Fielding, R., Irvine, U., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, RFC Editor.

[Hoffman 2013] Hoffman, P. (2013). O Tao do IETF. RFC 00, RFC Editor.

[Kaduk et al. 2014] Kaduk, B., Schaad, J., and athleen Moriarty (2014). Authentication and Authorization for Constrained Environments. RFC 00, RFC Editor.

[Navas et al. 2016] Navas, R. E., Lagos, M., Toutain, L., and Vijayasankar, K. (2016). Nonce-based authenticated key establishment over oauth 2.0 iot proof-of-possession architecture. *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 317–322.

[P. Eronen and H. Tschofenig 2005] P. Eronen, E. and H. Tschofenig, E. (2005). Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC 4279, RFC Editor.

[P. Wouters et al. 2014] P. Wouters, E., H. Tschofenig, E., Gilmore, J., Weiler, S., and Kivinen, T. (2014). Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). RFC 7250, RFC Editor.

[Postel 1980] Postel, J. (1980). User Datagram Protocol. RFC 768, RFC Editor.

[R. Fielding and J. Reschke 2014] R. Fielding, E. and J. Reschke, E. (2014). Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. RFC 7230, RFC Editor.

[Rescorla and Modadugu 2012] Rescorla, E. and Modadugu, N. (2012). Datagram Transport Layer Security Version 1.2. RFC 6347, RFC Editor.

[Rosenberg et al. 2012] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E. (2012). SIP: Session Initiation Protocol. RFC 3261, RFC Editor.

[Schaad and Cellars 2017] Schaad, J. and Cellars, A. (2017). CBOR Object Signing and Encryption (COSE). RFC 8152, RFC Editor.

[Seitz et al. 2016] Seitz, L., Gerdes, S., Selander, G., Mani, M., and Kumar, S. (2016). Use Cases for Authentication and Authorization in Constrained Environments. RFC 7744, RFC Editor.

[Shelby et al. 2014] Shelby, Z., Hartke, K., and Bormann, C. (2014). The Constrained Application Protocol (CoAP). RFC 7252, RFC Editor.

[Simmonds et al. 2004] Simmonds, A., Sandilands, P., and Van Ekert, L. (2004). An ontology for network security attacks. In *Asian Applied Computing Conference*, pages 317–323. Springer.

[Stallings 2014] Stallings, W. (2014). *Cryptography and Network Security: Principles and Practice, International Edition: Principles and Practice*. Pearson Higher Ed.

[T. Bray 2014] T. Bray, E. (2014). The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159, RFC Editor.

[Vasseur 2014] Vasseur, J. (2014). Terms Used in Routing for Low-Power and Lossy Networks. RFC 7102, RFC Editor.