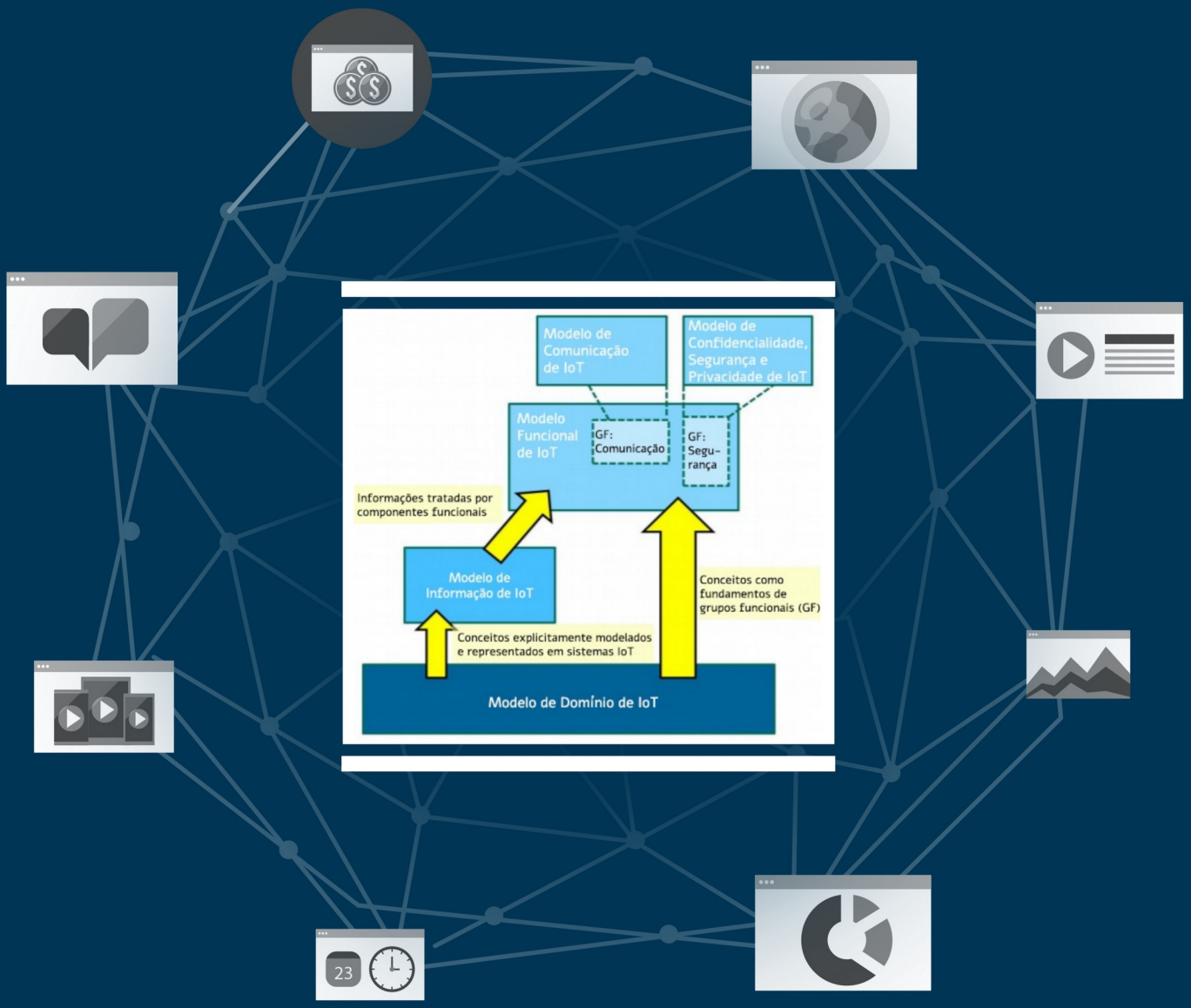
Universidade Federal da Paraíba | Departamento de Informática
Volume 5 Número 2 - Dez//2021

Revista Comunicações em Informática



ISSN: 2595-0622

http://periodicos.ufpb.br/index.php/cei

Avaliação Experimental da Plataforma Archive Fixity Anchor em Cenários de Adulteração de Dados em Ambiente Simulado de Repositório Digital Confiável

Filipy Galiza e Rostand Costa

Programa de Pós-Graduação em Informática (PPGI), Laboratório de Aplicações de Vídeo Digital (LAVID)
Centro de Informática – Universidade Federal da Paraíba (UFPB)
filipy@uepb.edu.br, rostand@lavid.ufpb.br

Resumo: Para auxiliar os Repositórios Digitais Confiáveis (RDC) em sua missão de garantir a fixidez dos documentos arquivísticos digitais, uma plataforma denominada Archive Fixity Anchor pode ser utilizada concomitantemente para que, a partir do uso combinado de Árvores de Merkle e *blockchains*, a Plataforma possa fornecer mecanismos confiáveis para o registro e auditoria da fixidez dos acervos nos repositórios. Este trabalho visa documentar a avaliação do comportamento desta plataforma e de um software de repositório no tratamento da fixidez de um acervo de objetos digitais submetidos a cenários simulados de adulteração de dados.

Palavras-chave: *preservação digital; fixidez; RDC; blockchain*.

1. Introdução

Sistemas de *software* tidos como referência para construção de repositórios digitais [1], como o Archivematica [2] e o RODA [3] executam a verificação da fixidez (*fixity check*) dos objetos digitais sob seus domínios com base em procedimentos padrão: comparando informações de fixidez (*fixity information*) geradas instantaneamente a partir dos objetos com *valores* de referência previamente registrados localmente referentes a esses objetos.

Esses processos de verificação normalmente utilizados podem ser insuficientes para garantir a fixidez em um Repositório Digital Confiável (RDC) frente as ameaças que as informações digitais estão expostas, principalmente no longo prazo, devido a degradação ocasional [4] ou adulteração intencional dos dados [5]. Portanto, a vulnerabilidade dos valores de referência para o controle de fixidez põe em risco a confiabilidade do acervo preservado, pois a desconfiança dessas informações pode fazer com que um acervo adulterado seja considerado legítimo, assim como a confiança em um acervo legítimo pode ser perdida.

Diante essa preocupação, Galiza e Costa [6] propõem uma abordagem baseada no uso de Árvores de Merkle e *blockchains* para que sejam gerados e armazenados imutavelmente valores de referência para o controle de fixidez dos acervos em RDC, além dos recursos de redundância utilizados pelos repositórios. A partir dessa proposta foi desenvolvida uma plataforma denominada Archive Fixity Anchor (AFA), que implementa os principais mecanismos para apoiar os conceitos defendidos pelos autores.

Os códigos e informações adicionais acerca da Plataforma desenvolvida podem ser encontrados em [7] e no seu atual estágio de desenvolvimento permite registrar manualmente os Pacotes de Arquivamento de Informações (Archival Information Packages – AIP) de um acervo digital com seus identificadores associados e viabiliza a auditoria da fixidez desses AIP a partir da consulta de valores confiáveis de referência ancorados em *blockchain*.

Este trabalho visa apresentar os resultados obtidos a partir da avaliação experimental da Plataforma em um ambiente simulado de RDC, montado utilizando os elementos mínimos necessários a essa tarefa, resumido no uso da AFA e do Archivematica como *software* de repositório digital, assim como todo o arcabouço tecnológico necessário a correta execução desses elementos.

Como avaliação experimental foram simulados três possíveis cenários considerados passíveis de acontecimento na prática de um repositório digital, onde se tenta simular algum tipo de alteração maliciosa no acervo, visando em todos os cenários avaliar o comportamento do *software* de repositório Archivematica e da plataforma AFA no âmbito da verificação de fixidez.

A primeira seção deste trabalho realiza uma contextualização acerca da temática abordada; na segunda seção se apresenta resumidamente a metodologia adotada na execução da avaliação; na seção 3 são apresentados os experimentos realizados e os resultados obtidos; e na seção 4 são feitas discussões e considerações finais.

2. Metodologia

Para a execução da avaliação, um ambiente de repositório digital foi simulado utilizando o *software* Archivematica, escolhido como *software* de repositório de acordo com as recomendações de Rodrigues [1] para tal. Como esperado, também compõe o ambiente o *software* AFA, alvo da avaliação deste trabalho e que visa apoiar as ações relacionadas a fixidez nesse repositório.

Esse ambiente de experimentos foi montado seguindo as recomendações de *hardware* e *software* necessários para a correta execução dos sistemas escolhidos de acordo com suas documentações [8][9], seguindo a instalação e configuração padrão ou recomendada para os sistemas e que resultou em um ambiente de repositório funcional e apto a ingestão de objetos para preservação.

Além da correta execução dos sistemas, se faz necessário o registro de objetos digitais no Archivematica para que esses sejam transformados em pacotes aptos para a preservação (AIP).

Como objetos digitais, foram selecionados documentos de imagem disponíveis em [7], que são registrados no Archivematica de acordo com as instruções documentadas em [8] na medida em que se executa os cenários de simulação.

Após os registros, os pacotes resultantes podem ser consultados pela aba “Packages” do Archivematica Storage Service, onde são apresentadas informações como o identificador único (UUID) e o local atual de armazenamento (caminho no sistema) para se ter acesso ao pacote, informações necessárias ao registro do mesmo na AFA, realizados de acordo com as instruções documentadas em [7] no decorrer da execução dos cenários de simulação.

O tratamento dos objetos e sua transformação em AIP são conduzidos automaticamente pelo Archivematica que, por padrão, permite o acompanhamento e possíveis intervenções pelo usuário, dentre as quais estão algumas opções de personalização para adequação aos padrões e objetivos de preservação do repositório. Mas para o registro dos objetos aqui relatados, além de não adicionar metadados durante esses processos iniciais, tais opções de intervenções foram definidas com valores recomendados pela documentação do Archivematica ou, na ausência de recomendação, definidas com as opções que resultassem em menos etapas de processamento.

Em cada um dos cenários de avaliação acontecem os processos de registro de objetos, a verificação da fixidez desses, a adulteração de dados e posterior repetição da conferência da fixidez, para, com isso, observar o comportamento dos *softwares* com relação a identificação de inconsistências nos AIP nos cenários avaliados.

Se espera que a AFA consiga identificar inconsistências de fixidez mesmo nos cenários em que o Archivematica possa não identificar, podendo demonstrar seu potencial incremento nas ações de preservação dos repositórios no aspecto de verificação e garantia da fixidez dos acervos.

Para a verificação da fixidez pelo Archivematica será utilizado um mecanismo nativo para esse controle, que acompanha o Archivematica Storage Service e que está disponível através de um API *endpoint* com funcionamento e uso documentado em [9]. Para facilitar sua referência no texto, esse método de verificação será referenciado apenas como *fixity endpoint*.

Na AFA, a verificação de fixidez acontece através do recurso de auditoria e seu uso está documentado em [7].

3. Avaliação e Resultados

Os três cenários de avaliação experimental e os resultados encontrados a partir deles são documentados nesta seção. Neles, foram apenas simuladas adulterações consideradas maliciosas e não desejadas para acontecimento em um ambiente de RDC.

3.1 Cenário 1: Adulteração simples de um pacote de informação

Nesse cenário se visou avaliar o comportamento do Archivematica e da AFA frente a simulação de uma possível adulteração de pacotes de informação, que poderia acontecer de forma maliciosa ou ocasional.

Para a simulação deste cenário foram registrados quatro objetos digitais no Archivematica, o que resultou na criação de quatro AIP, os quais foram devidamente registrados na AFA.

Após a confirmação do estado consistente da fixidez em ambos os sistemas, um dos pacotes foi escolhido aleatoriamente e teve seu conteúdo intencionalmente modificado, mas foi adulterado de forma a permanecer com as mesmas informações do pacote original numa tentativa de passá-lo como autêntico e inalterado nos processos de verificação de fixidez, simulando uma corrupção maliciosa.

Após efetivado a alteração no pacote, uma nova checagem de fixidez foi requisitada através da consulta ao *fixity endpoint* e da auditoria na AFA para se obter o estado atualizado de fixidez desse pacote, sabendo que esse não corresponde mais ao seu estado original.

A ferramenta do Archivematica detectou a inconsistência do pacote e informou sobre o estado de falha da fixidez desse, ainda fornecendo algumas informações adicionais sobre o problema (relatando quantos arquivos e o tamanho total em *bytes* esperado e o que foi constatado).

De maneira semelhante, a verificação de fixidez do pacote na AFA também alertou sobre a inconsistência de sua fixidez, mas diferente do Archivematica, a AFA não fornece detalhes sobre a inconsistência do registro. Isso acontece devido as diferentes formas de implementação do controle da fixidez, que no Archivematica se baseia no uso das especificações do formato Bagit [10] para guiar a formação dos pacotes, o que inclui informações adicionais sobre seu conteúdo, enquanto a AFA abstrai as informações do conteúdo e apenas considera as informações de *hash*.

Entretanto, independente dos detalhes fornecidos, neste cenário, tanto o Archivematica quanto a AFA alertaram corretamente para a falha da fixidez do pacote adulterado, ficando o repositório munido de duas ferramentas que lhe garante a correta detecção de corrupção de um pacote, como o simulado nesse cenário.

3.2 Cenário 2: Adulteração de um pacote de informação e de suas respectivas informações de fixidez

Nesse cenário se visou avaliar o comportamento do Archivematica e da AFA frente a simulação de uma possível adulteração puramente maliciosa de um pacote de informação, onde um atacante tentaria forjar a autenticidade de um pacote alterado.

Para a simulação deste cenário foram registrados outros quatro objetos digitais no Archivematica, o que resultou na criação de mais quatro AIP, cujos foram devidamente registrados na AFA.

Após a confirmação do estado consistente da fixidez em ambos os sistemas, um dos pacotes foi escolhido aleatoriamente e teve seu conteúdo intencionalmente modificado, mas foi adulterado de forma a permanecer com as mesmas informações do pacote original numa tentativa de passá-lo como autêntico e inalterado nos processos de verificação de fixidez, simulando uma corrupção maliciosa.

Além disso, nesse cenário o pacote foi reconstruído de acordo com as especificações do Bagit, com suas novas informações de fixidez com auxílio da ferramenta bagit-python, almejando o reconhecimento do pacote como autêntico pelo *fixity endpoint* do Archivematica.

Após a manipulação do pacote, uma nova checagem de fixidez foi requisitada através da consulta ao *fixity endpoint* e da auditoria na AFA para se obter o estado atualizado de fixidez desse pacote, sabendo que esse não corresponde mais ao seu estado original.

Nesse caso, a ferramenta do Archivematica considerou o pacote em questão como se permanecesse com sua fixidez consistente, ainda que o conteúdo do pacote estivesse alterado. Esse falso sucesso na verificação é mantido ao se obter detalhes no sistema sobre a verificação de fixidez do pacote, levando o usuário a crer que a fixidez do pacote continua inabalada.

Por outro lado, a verificação de fixidez do pacote na AFA retornou um alerta sobre a inconsistência de sua fixidez.

Essa divergência de resultados acontece devido as diferenças na implementação do controle da fixidez, como já previamente esclarecido e, nesse caso, mostrou a relevância do uso da AFA para apoiar o controle de fixidez em um repositório, que apenas com a ferramenta de verificação nativa do Archivematica pôde ser facilmente burlado nesse cenário.

3.3. Cenário 3: Adulteração de um pacote de informação e de suas respectivas informações de fixidez registradas na AFA

Nesse cenário se visou avaliar o comportamento da AFA frente a simulação de uma possível adulteração puramente maliciosa de um pacote de informação e das informações na AFA relacionadas a fixidez desse pacote, onde um atacante tentaria forjar a autenticidade de um pacote alterado manipulando os dados da AFA para que essa apoie sua ação.

Para esse cenário se optou por utilizar o mesmo conjunto de pacotes já previamente registrados nos sistemas no Cenário 2, dando continuidade aos processos iniciados nesse cenário anterior e eliminando a necessidade de refazer os mesmos processos a fim de se alcançar o estado no qual se findou o experimento documentado nesse último cenário.

Foi considerada, então, a escolha do mesmo pacote alvo do cenário anterior, onde esse foi devidamente manipulado e se conheceu o comportamento do *fixity endpoint* do Archivematica e da AFA frente a essa manipulação. Mas agora se pretende forçar a situação para que a AFA também apoie a legitimidade desse pacote adulterado.

Conhecendo o funcionamento da Plataforma, foi possível adulterar seus dados locais relacionados ao pacote alvo para que esse possa ter sua fixidez considerada consistente pela AFA.

Para a manipulação dos dados bastou estruturar as informações dos pacotes na mesma ordem em que se foram executados os registros legítimos, posicionando devidamente os novos valores maliciosos gerados a partir de uma nova Árvore de Merkle. No entanto, apesar da devida manipulação, ao se chamar uma auditoria na plataforma, essa continuou a alertar para a inconsistência nos dados relacionados a fixidez do pacote alvo.

Essa detecção de inconsistência é sentida pela plataforma devido à divergência entre as informações de fixidez (*hash* raiz das Árvores) armazenadas localmente e de seu valor âncora salvo na rede *blockchain*. Mas, ainda que se tente alterar apenas o valor *hash* raiz local para tentar coincidir com o valor âncora, a plataforma detecta que esse valor não condiz com o valor *hash* raiz gerado pela Árvore de Merkle que considere o *hash* do pacote adulterado e continua alertando sobre a inconsistência de fixidez dos dados relacionados ao pacote adulterado.

4. Discussões e Considerações Finais

Este trabalho se propôs a documentar a realização da avaliação experimental da plataforma desenvolvida Archive Fixity Anchor em cenários de adulteração de dados em ambiente simulado de RDC utilizando o *software* Archivematica.

A partir dos experimentos aqui documentados pôde se observar que a plataforma AFA se comportou de acordo com o esperado para sua versão implementada e satisfez a expectativa dos resultados: alertando sobre o estado genuíno de fixidez daqueles pacotes em si registrados e de suas próprias informações locais sobre os registros a partir do uso de um valor âncora confiável em *blockchain*, mesmo quando as inconsistências foram despercebidas pelo mecanismo de controle do Archivematica.

Se considera que uma das formas que um atacante pode ter para tentar contornar o correto resultado entregue pela Plataforma, a exemplo do resultado obtido no Cenário 3, seria o atacante descobrindo uma outra transação na mesma rede *blockchain* que carregue dados que represente o valor âncora necessário para fazer com que os dados locais adulterados possam parecer autênticos e, com isso, ele poderia ter um endereço de transação coerente para substituir nos dados locais da Plataforma e finalmente validar os registros adulterados. Porém, essa é uma condição improvável de se alcançar, principalmente se for considerado que em uma implementação ideal, a Plataforma prevê o uso de redes e algoritmos de *hash* redundantes, o que reduziria ainda mais a probabilidade do atacante coincidir todas essas variáveis a seu favor. Isso tudo sem considerar a marca temporal das transações, que a Plataforma não considera em suas análises de fixidez, mas que se vier a ser considerada de alguma forma, se torna mais uma variável a pesar na obstrução do ataque.

Uma outra forma mais simples de burlar o mecanismo de auditoria da AFA seria executar os registros normalmente informações derivadas de adulteração, obtendo um registro legítimo para informações adulteradas e a partir disso, tentar fazer um registro local se passar por outro. Esse ataque se mostra facilmente viável principalmente quando o registro alvo do ataque é relativamente recente, uma vez que a marca temporal pode não ser tão decisiva nesses casos. Porém, nesse último caso pode se pôr em discussão o interesse de ataques a registros recentes e para todos os casos pode se considerar a implementação de recursos que tratem as transações realizadas pela carteira do repositório na rede *blockchain* com fins de registro como referência para as transações registradas localmente, além da possibilidade de se considerar a marca temporal das transações como mais um elemento de controle.

Também se estima que uma outra forma de enganar o correto funcionamento da Plataforma e não considerada nos experimentos poderia ser a de adulteração de seu código fonte a fim de direcionar seu funcionamento em favor de um ataque. Para esses casos resta a realização de uma auditoria no código fonte da Plataforma em execução ou a obtenção, sempre que se considerar pertinente, dos códigos oficiais da Plataforma nos casos em que não se trabalha com versões modificadas dessa.

O fato da realização dos experimentos utilizando um número reduzido de amostras se deu pela necessidade de intervenção manual para registro na AFA e a racionalização do tempo desta pesquisa.

Apesar de não abordado nesses experimentos, o Archivematica dispõe de mecanismos para automatizar o registro de objetos digitais, mas a indisponibilidade desse tipo de recurso na AFA compromete atualmente o registro de grandes acervos (centenas ou milhares de pacotes) e consequentemente seu uso em produção.

Bibliografia

[1] Rodrigues, M.M. (2015) Repositório Arquivístico Digital Confiável para o Patrimônio Documental Oriundo do Processo Judicial Eletrônico. Dissertação de Mestrado. Programa de Pós-Graduação em Patrimônio Cultural. Universidade Federal de Santa Maria. Online: https://repositorio.ufsm.br/handle/1/11050.

[2] Archivematica: open-source digital preservation system. Online: https://www.archivematica.org/. Acesso em 18/01/2021.

[3] RODA - Repositório para preservação de informação digital - KEEP SOLUTIONS. Online: https://www.keep.pt/produtos/roda-repositorio-para-preservacao-de-informacao-digital/. Acesso em 18/01/2021.

[4] Wright, R.; Miller, A.; Addis, M. (2009) The Significance of Storage in the “Cost of Risk” of Digital Preservation. *International Journal of Digital Curation* 4(3): 104–122. DOI: 10.2218/ijdc.v4i3.125

[5] National Research Council (2005) Building an Electronic Records Archive at the National Archives and Records Administration: Recommendations for a Long-Term Strategy. National Academies Press, Washington, D.C. DOI: 10.17226/11332

[6] Galiza, F.; Costa, R. (2020). Uma Abordagem Baseada em DLTs para Garantia da Fixidez de Repositórios Digitais Confiáveis (RDCs). Anais do III Workshop em Blockchain Teoria, Tecnologias e Aplicações, pp. 41–54. SBC. DOI: 10.5753/wblockchain.2020.12432

[7] Plataforma para auxiliar os Repositórios Digitais Confiáveis a garantir a fixidez de seus acervos. Online: https://bit.ly/393zvol. Acesso em 22/07/2021.

[8] Archivematica Quick-Start Guide | Documentação (Archivematica 1.12.1) | Archivematica: open-source digital preservation system. Online: https://www.archivematica.org/pt-br/docs/archivematica-1.12/getting-started/quick-start/quick-start/. Acesso em 18/01/2021.

[9] Fixity | Documentation (Archivematica Storage Service 0.17.1) | Archivematica: open-source digital preservation system. Online: https://www.archivematica.org/en/docs/storage-service-0.17/fixity/. Acesso em 22/07/2021.

[10] The BagIt File Packaging Format (V1.0). Online: https://tools.ietf.org/html/rfc8493. Acesso em 18/01/2021.

Um Estudo Comparativo de Padrões de Arquiteturas para IoT e as Aplicações Comerciais

João Paulo Silvino Belo da Silva e Cleonilson Protásio de Souza

Programa de Pós-Graduação em Engenharia Elétrica
Centro de Energias Alternativas e Renováveis – Universidade Federal da Paraíba (UFPB)
{joaopaulo.silva, protasio}@cear.ufpb.br

Resumo: Existem hoje várias plataformas para Internet das Coisas *(IoT, Internet of Things)* com soluções diferenciadas, cabendo ao interessado avaliar as vantagens e desvantagens que cada empresa disponibiliza. Uma das formas para efetuar uma avaliação é a análise da arquitetura estabelecida e se seguem normas e documentos de padronização existentes. Neste contexto, este trabalho descreve um estudo, realizado por meio de pesquisa bibliográfica, sobre padrões arquiteturais em IoT, como IEEE Std 2413-2019, IoT-A e a IIRA. Como resultado, um estudo comparativo entre algumas arquiteturas de plataformas consolidadas no mercado, como Azure da Microsoft, e IBM Industry 4.0, e os padrões descritos é apresentado.

Palavras-chave: *IoT; padrões arquiteturais; plataformas de desenvolvimento.*

1. Introdução

O termo Internet das Coisas ou IoT foi primeiramente citado por Kevin Ashton em 1999 em uma apresentação sobre aplicação da tecnologia RFID (*Radio Frequency Identification*) na cadeia logística da Procter & Gamble [1]. Por volta de 2008, o termo IoT passou a ter novo significado, sendo aplicado ao uso de várias tecnologias que utilizam a internet para interconectar dispositivos embarcados. Atualmente, o conceito de IoT tornou-se ainda mais amplo e se baseando em objetos do cotidiano que interagem e geram dados autonomamente via internet [2].

A indústria de IoT apresenta grande crescimento e atualmente já conta com mais de 20 bilhões de objetos conectados [2] muito devido à disponibilização no mercado de várias plataformas computacionais em IoT por grandes empresas de nuvem de alcance mundial.

Para garantir a compatibilidade, segurança e uniformidade das diferentes plataformas utilizadas em IoT, tornou-se bastante necessário adotar padrões e assim diferentes organizações passaram a propor e estabelecer padrões arquiteturais para implementação de plataformas em IoT.

Em 2014, o IEEE (*Institute of Electrical and Electronics Engineers*) definiu o conceito de IoT como simplesmente “uma rede de itens embarcados e com sensores conectados à internet” [3]. Entretanto, há um conjunto de dispositivos, como os *smartphones*, que possuem sensores embarcados, mas não estão no contexto de IoT.

Observa-se então a dificuldade em uniformizar conceitos em IoT iniciando pelo próprio termo IoT dado que este envolve um conjunto de diversas tecnologias aplicadas em diferentes áreas e contextos e que possuem, como principal característica, a conexão de objetos ou coisas com a internet.

Outros conceitos também existem. Por exemplo, um conceito mais amplo de IoT, apresentado em [4], seria: “Um sistema de entidades (incluindo dispositivos físico-cibernéticos, pesquisas de dados e pessoas) que trocam informações e interagem com o mundo físico”.

Portanto, foi a partir da necessidade de uniformidade nas definições relacionadas a IoT, bem como suas aplicações, características e estruturas, que foram iniciados o estabelecimento de processos de padronização e os decorrentes padrões de mercado.

Em 2015, o IEEE foi iniciada a concepção de um padrão de *framework* arquitetural para IoT, finalizado em 2019: o padrão IEEE Std 2413-2019. Como ponto de partida, este padrão caracteriza uma coisa em IoT como “um componente ou sistema que possui funções, propriedades e meios de comunicação”, conforme Figura 1. Assim, para um dispositivo ser considerado uma “coisa” em IoT, este necessita desempenhar funções específicas, conter propriedades que caracterizem interna ou externamente o ambiente ao seu entorno e ter capacidade de trocar informações com aplicações e serviços através da *internet*.

Figura 1. Uma “coisa” conforme o padrão IEEE Std 2413-2019 [4].

Além do padrão IEEE Std 2413-2019, existem outros padrões que abordam arquiteturas em IoT, seus blocos constitutivos e suas funcionalidades, como a IoT-A (*Internet of Things - Architecture*) e a IIRA (*Industrial Internet Reference Architecture*).

Neste trabalho, serão apresentadas as principais características dos principais padrões atuais em arquitetura de plataformas em IoT e um estudo comparativo breve entre esses padrões com as

arquiteturas de plataformas de nuvem de grandes empresas, como a Intel e a IBM.

2. Metodologia

Este estudo foi desenvolvido tendo como base uma pesquisa bibliográfica sistemática partindo da busca por definições de IoT e conceitos associados, seguindo-se da análise de normas e padrões pertinentes às arquiteturas de sistemas no contexto de IoT. A partir do resultado da pesquisa bibliográfica, buscou-se averiguar o emprego das normas e padrões encontrados com plataformas de computação em nuvem para IoT de grandes empresas do mercado. Por fim, foi realizado um estudo comparativo entre as padronizações em IoT levantadas e as plataformas comerciais.

3. Arquiteturas de Referência

Adotar uniformidade por meio de um padrão em IoT, pode proporcionar a superação de alguns desafios dentre os quais: a identificação e endereçamento das coisas mesmo com uso massivo; promover a interoperabilidade dos sistemas; e fornecer gerenciamento com segurança e privacidade para usuários e empresas [5].

As principais arquiteturas de referência encontradas e orientadas especificamente para IoT são as seguintes:

- IEEE Std 2413-2019 (*Standard for an Architectural Framework for the Internet of Things);*
- IoT-A (*Internet of Things - Architecture*); e
- IIRA (*Industrial Internet Reference Architecture*).

Estes padrões arquiteturais foram desenvolvidos considerando diferentes abordagens e priorizando aspectos variados, mas, em comum, visam estabelecer uniformidade e a base necessária para o desenvolvimento, a concepção, a operação e a manutenção de soluções em IoT. A concepção das arquiteturas de referência também é baseada em padrões previamente existentes, que orientam os procedimentos para elaboração e estabelecem suas características.

O padrão IoT-A e o IEEE Std 2413-2019, e indiretamente o IIRA, são baseadas no padrão ISO/IEC/ IEEE 42010:2011 *Systems and software engineering*, que estabelece como as especificações arquiteturais dos sistemas devem ser organizados e descritos. A seguir serão descritas as principais características desses padrões em IoT.

3.1 IEEE Std 2413-2019

O padrão IEEE Std 2413-2019 tem como característica principal reunir as particularidades relacionadas a cada um dos domínios de interesse (*domains*) de IoT, entre esses, *smart buildings*, *smart cities*, smart transportation, smart grid e healthcare.

O modelo conceitual do *framework* arquitetural adotado pela IEEE Std 2413-2019, que foi baseado no padrão ISO/IEC/IEEE 42010:2011, é apresentado na Figura 2, em que é possível observar as relações estruturais entre os *stakeholders* (parte interessada), seus *concerns* (interesses) e seus *domains* (domínios) e suas interrelações com a perspectiva de *framework* (estrutura arquitetural) e *viewpoint (perspectiva arquitetural).*

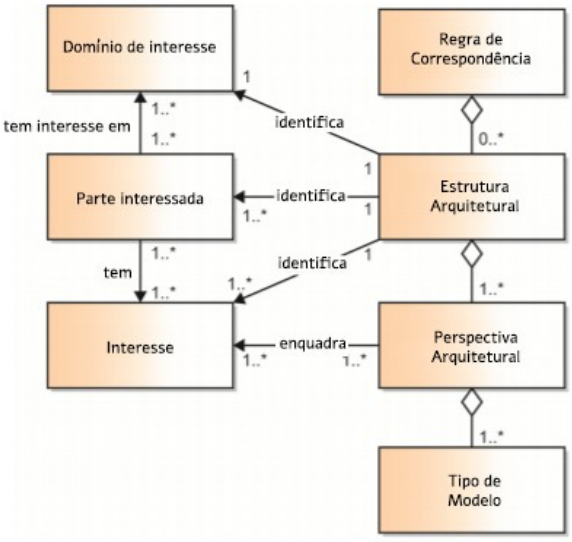


Figura 2. Modelo de f*ramework* arquitetural adotado pela IEEE Std 2413-2019 [4].

O *framework* visto na Figura 2 representa uma descrição completa da arquitetura de referência para uma dada aplicação. O *stakeholder* é o conjunto de indivíduos ou entidades que têm interesse ou relação com o sistema, a saber: os desenvolvedores, os usuários, os vendedores, os órgãos de regulação governamentais, entre outros e, por sua vez, os interesses (*concerns*) são as preocupações/necessidades de cada *stakeholder*. Os *viewpoints* são representações arquiteturais que relacionam os *stakeholders* e seus respectivos *concerns.* Exemplos de *viewpoints* são: *viewpoint* funcional, *viewpoint* de implementação, *viewpoint* de negócio (*business viewpoint)*, entre outros. Os modelos (*models*) são as representações mais próximas da realidade de um processo, conceito ou do dispositivo real. As regras de correspondência estabelecem a relação entre as *viewpoints* que, com os tipos de modelos (*model kinds*), consistem em elementos que sub-categorizam cada *viewpoint*, formam a base do *framework* arquitetural.

3.2 IoT-A (Internet of Things - Architecture)

O padrão IoT-A foi desenvolvido em 2013 por um grupo de empresas em conjunto com a *European Lighthouse*, projeto ligado à União Europeia. O padrão IoT-A foi um modelo de referência pioneiro já que naquela época ainda não era conhecida outra arquitetura relacionada à IoT e foi aplicada em diferentes domínios.

O padrão IoT-A apresenta perspectiva voltada aos aspectos relacionados à informação e à funcionalidade, abordando também requisitos técnicos [6]. Na Figura 3, é ilustrado o modelo de referência do padrão *IoT-A*, constituído de submodelos, que são as diferentes categorias que fundamentam a arquitetura.

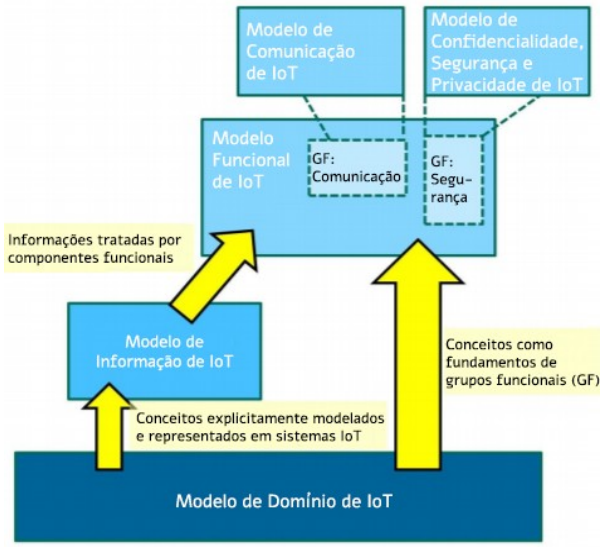


Figura 3. Interação entre submodelos na arquitetura IoT-A [7].

3.3 IIRA (Industrial Internet Reference Architecture)

O padrão IIRA foi desenvolvido especificamente para aplicações industriais, pois o interesse é na Internet das Coisas Industriais (IIoT) [6]. O padrão IIRA foi construído pelo *Industrial Internet Consortium* (formado por pesquisadores, empresas, universidades e órgãos governamentais) e finalizado em 2015.

Na Figura 4 é observado um dos modelos de arquitetura no padrão IIRA, baseado em três camadas (*three-tier*). Nela cada elemento do sistema no padrão IIRA é representado por um bloco funcional localizado em um dado nível (*tier)*, de acordo com sua funcionalidade. No nível de borda (*Edge Tier*), estão os dispositivos (coisas) e *gateways*. No nível de plataforma (*Platform Tier*), são executados serviços, isto é, neste nível ocorre recebimento, processamento e envio de dados que devem trafegar entre os dispositivos e as aplicações. Os blocos referentes às aplicações que recebem dados de dispositivos e enviam comando situam-se no nível de empreendimento (*Enterprise Tier*).

No padrão de 2008 foi adicionado a linguagem o que se chama de *coarray*, que é uma estrutura de dados que pode ser compartilhada entre diferentes *images*, sendo *images* cópias idênticas do executável. Essa característica permite que a linguagem se aproveite da computação paralela através de uma técnica de paralelismo chamada SPMD, do inglês *Single Program Multiple Data*. Em outras palavras, o objetivo era tornar a programação paralela de fácil implementação, fazendo com que várias *images* pudessem trabalham em diferentes dados ou partes de um mesmo *coarray*, fornecendo ao programador a possibilidade de trabalhar com programação de memória compartilhada.Além disso, foi introduzida também a estrutura *DO...CONCURRENT* que especifica laços sem interdependência, que é particularmente interessante para compiladores de paralelização automática.

Considerando somente as maiores empresas que disponibilizam plataformas em nuvem de IoT em um cenário mundial - a Azure da Microsoft, IBM Industrie 4.0, Intel IoT e a Amazon - observa-se que essas possuem arquiteturas próprias em IoT e oferecem serviços variados com intuito de atender domínios e necessidades diferentes e específicas dos clientes.

No levantamento realizado nos documentos oficiais dessas empresas, não foi possível associar as arquiteturas das plataformas Intel IoT e Amazon Web

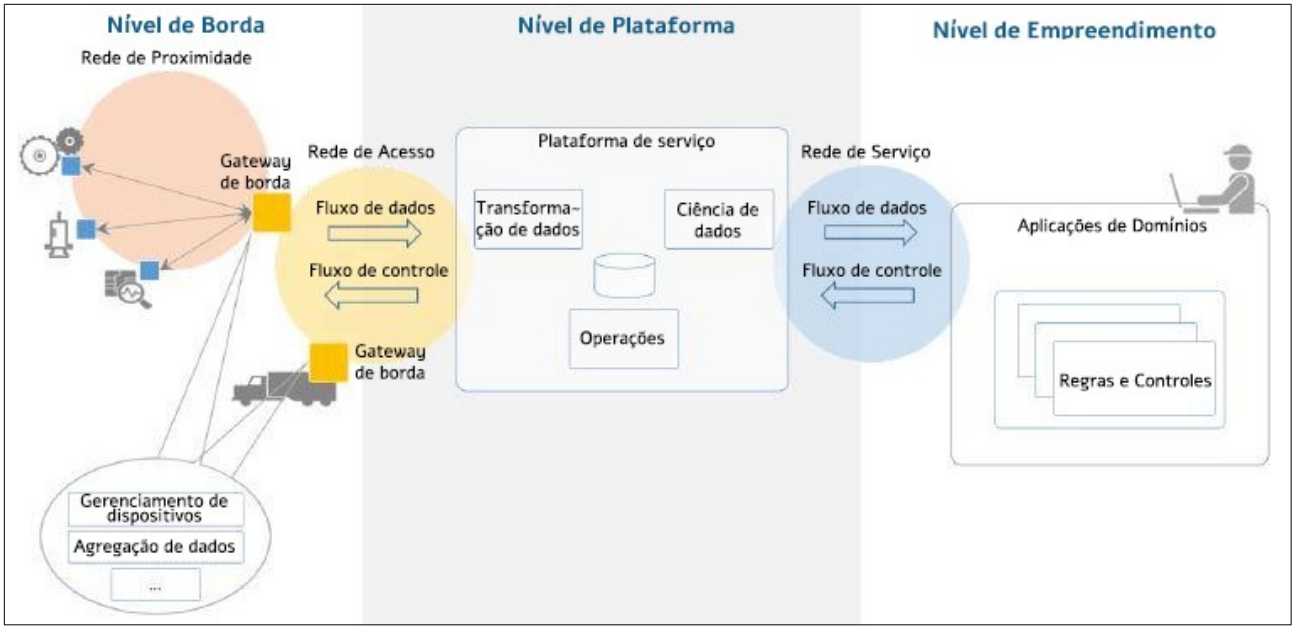


Figura 4. Modelo de arquitetura padrão three-tier do IIRA [8].

Services lambda, com as Arquiteturas de Referência apresentadas neste trabalho, pela falta de documentação. Para a Microsoft Azure e a plataforma IBM Industrie 4.0, a seguir será descrito um estudo comparativo.

Microsoft Azure: pelo levantamento bibliográfico realizado, a arquitetura de referência da Microsoft Azure não menciona diretamente a utilização de qualquer padrão na sua concepção, mas por estudo comparativo é possível associar a arquitetura Azure com o padrão IIRA, como visto na Figura 5, em que as colunas Coisas *(Things)*, *Insights* e Ações *(Actions)* podem ser facilmente associadas ao modelo *Three-tier* do padrão IIRA.

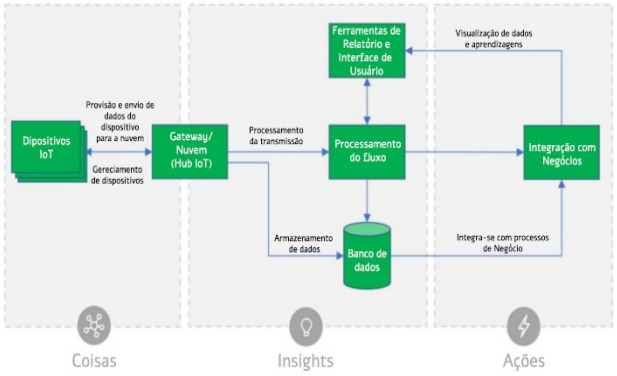


Figura 5. Arquitetura de Referência da Microsoft Azure IoT [9].

IBM Industrie 4.0: é caracterizada como uma versão industrial de serviços de IoT e é também baseada no *IIRA* [10]. Esta associação é evidenciada na utilização do padrão *Three-tier* do padrão IIRA, como pode ser visto na Figura 6.

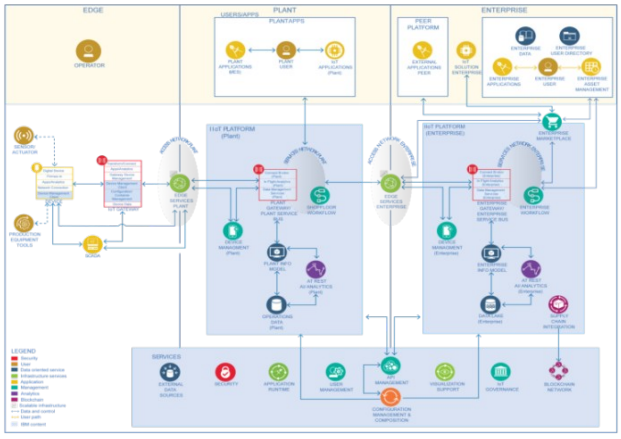


Figura 6. Arquitetura de Referência da *IBM Industrie 4.0* [11].

5. Conclusões

Neste trabalho foram apresentados os padrões principais de arquiteturas em IoT: o padrão IEEE Std 2413-2019, o IoT-A e o IIRA. Apresentou-se também suas principais características e uma breve comparação com as arquiteturas de plataformas de grandes empresas, como a Microsoft, a IBM, a Intel e a Amazon. Concluiu-se que existe um certo nível de associação entre os padrões e algumas dessas arquiteturas. Entretanto, não é simples determinar se uma dada plataforma se baseia ou não em um algum padrão, por meio somente da documentação oficial das arquiteturas comerciais.

Bibliografia

[1] That “Internet of Things” Thing. RFiD Journal. Online: https://www.rfidjournal.com/that-internet-of-things-thing Acesso em: 18 mai. 2021.

[2] Souza, C. P.; Baiocchi, O. (2018) Energy Resources in Agriculture and Forestry: How to be Prepared for the Internet of Things (IoT) Revolution, Energy Systems and Environment, Pavel Tsvetkov. IntechOpen, DOI: 10.5772/intechopen.74940

[3] Minerva, R.; Biru, A.; Rotondi, D. (2015) Towards a definition of the Internet of Things (IoT). 1ed. IEEE.

[4] IEEE Std 2413-2019: Standard for an Architectural Framework for the Internet of Things (IoT). (2020). DOI: 10.1109/IEEESTD.2020.9032420

[5] Stankovic, J. (2014) Research Directions for the Internet of Things. *Internet of Things Journal* 1(1): 3-9. IEEE. DOI: 10.1109/JIOT.2014.2312291

[6] Weyrich, M.; Ebert, C. (2016) Reference Architectures for the Internet of Things. *IEEE Software* 33(1): 112-116. DOI: 10.1109/MS.2016.20

[7] Bauer, M. et al. (2013) Internet of Things – Architecture IoT-A Deliverable D1.5 – Final architectural reference model for the IoT v3.0.

[8] Industrial Internet Reference Architecture. Industrial Internet Consortium (IIC). Online: https://www.iiconsortium.org/IIRA.htm. Acesso em: 18 mai. 2021.

[9] Microsoft Azure IoT Reference Architecture. Online: https://azure.microsoft.com/pt-br/resources/microsoft-azure-iot-reference-architecture/. Acesso em: 18 maio 2021.

[10] Simplify the development of your IoT solutions with IoT architectures. Online: https://developer.ibm.com/technologies/iot/articles/iot-lp201-iot-architectures/. Acesso em 18 maio 2021.

[11] IBM Industrie 4.0 Reference Architecture. Online: https://www.ibm.com/cloud/architecture/architectures/iot_industrie_40. Acesso em: 18 maio 2021.

A DDoS Detection Algorithm Based on Chaos Using Density of Maxima

Josimar Tavares, Ewerton M. Salvador, Alisson V. Brito

Centro de Informática – Universidade Federal da Paraíba (UFPB)
josimartaf@gmail.com, ewerton@ci.ufpb.br, alisson@ci.ufpb.br

Abstract: In this paper, a technique is presented to analyze and detect anomalies in network data flows. The Density of Maxima is used to measure the chaotic behavior of network traffic in order to detect DDoS attacks. The experiments were performed using data containing synthetic traffic created with the JMeter tool, real data from the World Cup 1998, CAIDA 2007 and DARPA 2009 DDoS Malware datasets. All tests demonstrated the effectiveness of the technique in identifying when the attacks occurred, as well as in separating regular from DDoS traffic.

Keywords: *network traffic analysis; chaos theory; density of maxima; network security.*

1. Introduction

Among the security issues of the Internet, one that stands out is the Distributed Denial of Service (DDoS) attack. This kind of attack aims to disrupt services on the victims, partially or entirely preventing them from functioning. According to the Kaspersky Lab's Global IT Security Risk Survey of 2017, 50% of enterprises claim that the frequency and complexity of DDoS attacks targeting organizations are growing every year [1].

Attacks of this type are usually related to the use of botnets, which are the primary tools for conducting DDoS attacks. Nowadays, the traffic generated by DDoS attacks has reached the terabytes scale. Three of the major DDoS attacks known to date are the following: the attack against Google, which registered a traffic of 2.54 Tbps; the attack against the Microsoft Azure Cloud, which peaked 2.4Tbps of traffic; and the attack against Amazon, which registered 2.3Tbps of traffic [2].

Among other relevant DDoS incidents, one can cite the attack against the GitHub website, which had a traffic record of 1.3 Terabits per second (Tbps) [3]. The Mirai Internet of Things (IoT) botnet attack occurred in 2016 and exceeded 600 Gbps of traffic [4]. Renowned companies (e.g. Twitter, Spotify, and Amazon) also had problems because of DDoS attacks on their services for almost two hours on Oct 21, 2016. The revenue loss due to DDoS attacks has touched to $209 million in the first quarter of 2016, compared to $24 million for all of 2015 [5].

In addition to causing immediate and visible operational problems, many companies also claim that DDoS attacks are being used to cover up other types of incidents, leading to severe financial damage and reputation. Respondents claimed that DDoS attacks were serving as a smokescreen to cover up other attacks such as malware infection, data leakage, network intrusions, and financial thefts [1].

One of the main difficulties in dealing with DDoS attacks is that the attacking devices usually have fake IP and MAC addresses (spoofed), hiding the source of the disruption. Because of this enormous amount of connections, it is challenging to differentiate legitimate users from connections generated by attackers.

Several difficulties need to be handled when the DDoS and regular traffics are mixed. Non-standard ports, disguised ports, and network address translation (NAT) additionally increase the difficulties during classification [6].

This problem motivated new approaches to tackle these types of attacks, such as network traffic analysis together with other tools like Chaos Theory and Time Series. Some research approach methods that use DDoS attack detection implemented software and hardware platforms with Field Programmable Gate Arrays (FPGA). This type of device requires less than one microsecond to classify a sample of incoming traffic as an attack or a regular one [7].

The literature points out that a signal related to network traffic presents a chaotic behavior [8]. Thus, this chaotic behavior can be used to detect DDoS attacks [9-10], or to implement network intrusion detection systems [11]. Some works use temporal series analysis [12], while others use artificial neural networks [13]. All of those works use the Lyapunov Exponent technique to measure the chaotic behavior for identification of anomalies in network traffics.

In this work, a technique called Network Analysis based on Chaos using Density of Maxima (NAC-DM) is presented. NAC-DM is based on the fact that the density of peaks of a signal is related to its chaotic behavior [14]. Thus, a computationally simple approach that counts the number of peaks per time is successfully applied to detect DDoS attacks in network traffics.

2. Related Work

Currently, there are already in the literature some works that use chaos characterization technique to detect DdoS attacks. These papers are briefly presented in this section.

Khan, Ferens and Kinsner [11] apply chaos theory to measure the complexity of Internet packages in order to determine whether they are regular or anomalous. The work [9] uses an algorithm for detecting DDoS attacks using the ARIMA (Autoregressive Integrated Moving Average). This algorithm requires the IP address of the devices and the number of transmitted packets per minute. Chonka, Singh, and Zhou [8] have developed an algorithm that uses network self-similarity theory to

differentiate DDoS flood attack traffic from legitimate network traffic.

Wu and Chen propose an improvement to the technique introduced in a previous work [13]. The DDoS detection algorithm is improved based on the NADA (Network Anomaly Detection Algorithm).

Differently from the strategy adopted in this research, these related works use the Lyapunov Exponent technique along with other techniques, such as neural networks or time series models.

3. Experiments

Some data sets were used to validate the NAC-DM technique: (1) synthetic traffic using JMeter [15], which is a traffic-generating tool, (2) Fifa World Cup 98 data, and public datasets with DDoS attacks, such as (3) CAIDA 2007, and (4) DARPA 2009 DDoS Malware.

3.1 Synthetic Traffic with JMeter

The primary purpose of the JMeter tool was to generate HTTP traffic in a controlled manner [15]. The experiment allowed to explore different scenarios than the ones presented in third parties data sets. Using JMeter, it is possible to configure the number of users, data traffic, and specify attacks.

Another experiment simulating traffic on a Web server under DDoS attack was performed using Jmeter [15]. 2-hour traffic was generated with five simulated users performing HTTP requests to the webserver. Another device was used to perform DDoS attacks on the same server. Even though it is only one device, a large number of requests were made in comparison to regular traffic, as one can observe in Figure 1(a).

During the traffic generation, two attacks lasting 5 minutes each were configured against the webserver. The first attack occurred at 1800 seconds after the beginning of the traffic generation, and the second attack started after 4200 seconds. The values computed for NAC-DM are presented in Figure 1(b). There are two signals in the same plane, the first one (in red) represents the original signal plus the two attacks and the second one (in blue) is the regular traffic excluding the attack signals. As we can see, the values between X=60 and X=69 represent the interval between the beginning and the end of the first attack. Likewise, the values between X=141 and X=149 mark the beginning and the end of the second attack.

3.2 FIFA World Cup'98

1) Experiment 1

In this experiment, we analyzed real traffic data from FIFA World Cup ‘98, choosing May 3rd, 1998 (source: http://ita.ee.lbl.gov/html/contrib/WorldCup.html) as our analysis target.

A copy of the traffic generated by the DDoS attack was made, in order for it to be inserted into the original traffic at two different places. The original signal (blue), and the signal modified by the attack (red) are presented in the same chart (Figure 2(a)). The NAC-DM technique was used in both signals, and the obtained results can be seen in Figure 2(b). In this experiment, the attacks started at 7200 and 14400 seconds after the traffic started, and both lasted for about 5 minutes. The values of X = 240 and X = 481 correspond to the beginning of both attacks.

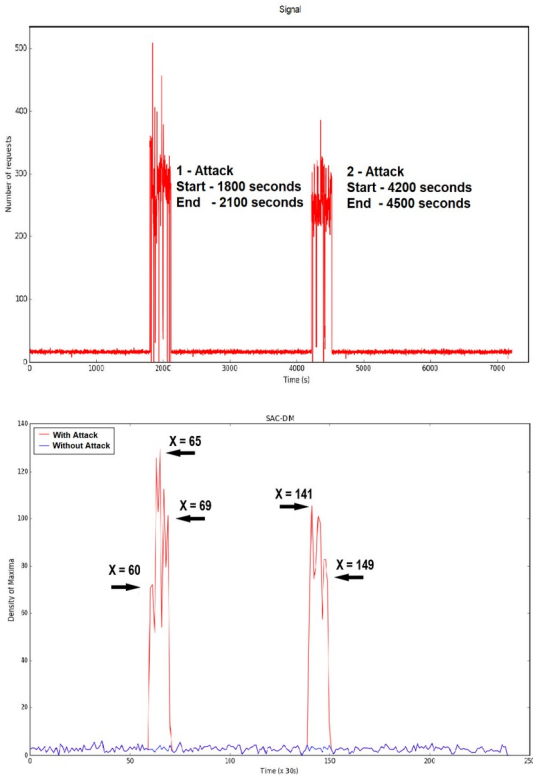


Figure 1. Signals in experiment with JMeter: (a) Original signal, (b) values of NAC-DM.

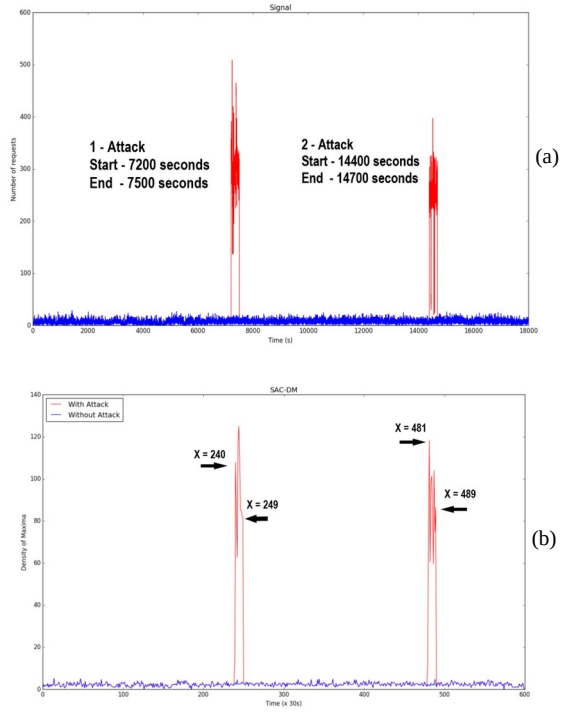


Figure 2. Results from World Cup, experiment 1: (a) Signal, (b) NAC-DM.

2) Experiment 2

This experiment aimed to observe the behavior of the algorithm in real traffic. Because of this, we looked for an interval in the traffic trace where access peaks occurred as the result of legitimate usage of the network. The selected day in the traffic trace was June 30th, 1998, where two decisive games at different times occurred. The second game had a high number of requests, but there were no attacks [16].

The results can be seen in Figure 3. The detection algorithm identified no attacks, which demonstrates that the technique is not only capable of detecting DDoS attacks, but also capable of not detecting false attacks.

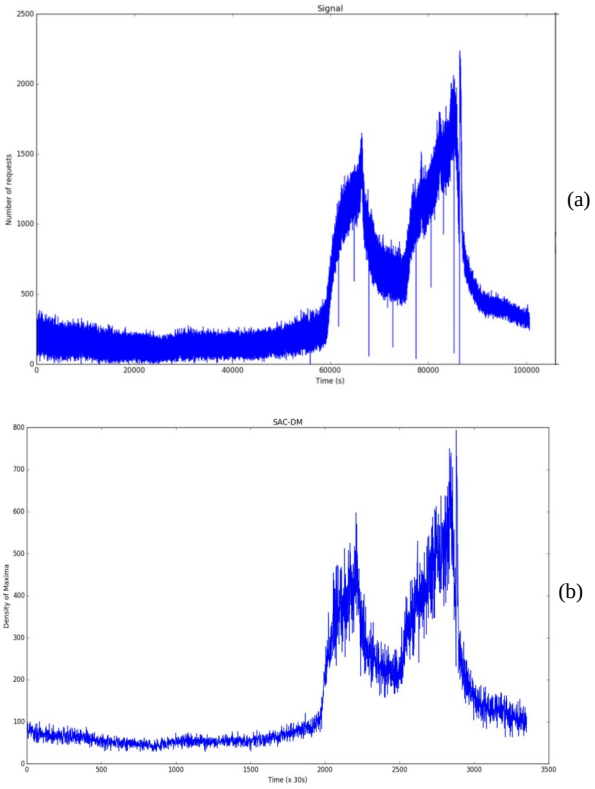


Figure 3. Results from World Cup, experiment 2: (a) Signal, (b) NAC-DM.

3.3 Datasets from CAIDA and DARPA

The datasets CAIDA 2007 [17] and DARPA 2009 DDoS Malware [18] are well known in the network analysis community. The dataset used by CAIDA 2007 in this experiment was the same used in [19]. The algorithm identified the beginning of the attack at X=54, as seen in Figure 4(a). This value corresponds to the value of X=1620, as seen in Figure 4(b).

Concerning the dataset from DARPA 2009, as it contains only 330 seconds, the sample size was configured to 5 seconds (instead of 30 as in previous experiments). The algorithm detected the attack at X=11 and at X =65, as seen in Figure 5(a). Both values correspond to the two attacks present on the signal, which started around 55 and 329 seconds, as seen in Figure 5(b). In other words, the NAC-DM showed peaks while the attacks occurred.

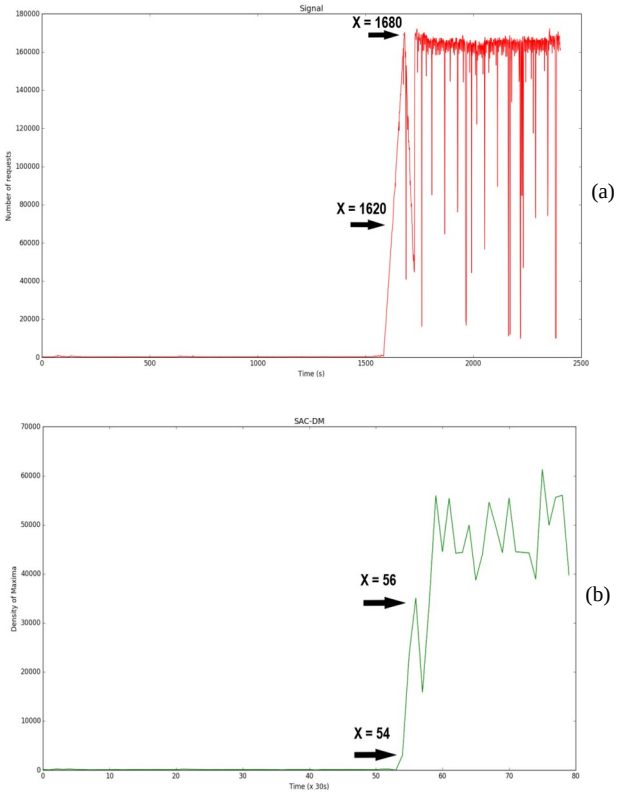


Figure 4. Analysis of dataset from CAIDA 2007: (a)Signal, (b)NAC-DM.

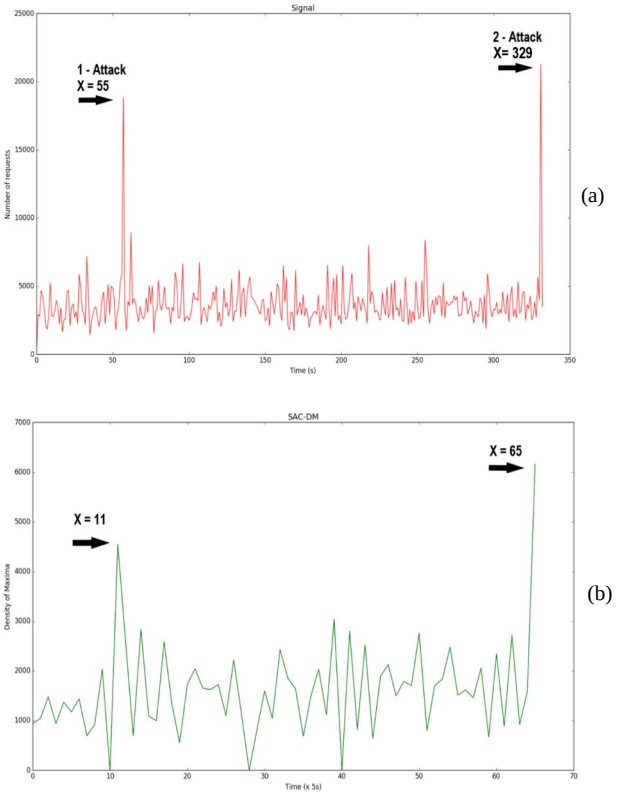


Figure 5. Analysis of dataset from DARPA 2009: (a)Signal, (b)NAC-DM.

4. Conclusion

In this work, a DDoS attack detection algorithm was presented based on the Chaos, using the Density of Maxima. The results obtained during the experiments were satisfactory, and confirmed the ability of the algorithm to detect attacks. In Table 1, the accuracy rate of the algorithm is presented. For all the scenarios, the accuracy rate was 100%.

One of the scenarios evaluated FIFA WC'98-2, a dataset without any DDoS attacks. However, there were two peaks due to a large number of access by legitimate users during the event. The algorithm did not detect any attacks on the dataset, which resulted in no false positive. It makes clear its capability to distinguish a legitimate peak of traffic from a peak caused by a DDoS attack.

Table 1: Accuracy of results from experiments.

Dataset	Attacks	Detection	Precision
JMeter	02	02	100%
FIFA WC'98 - 1	02	02	100%
FIFA WC'98 - 2	00	00	100%
CAIDA 2007	01	01	100%
DARPA 2009	02	02	100%

Bibliography

[1] K. Lab (2017) Kaspersky lab research shows ddos devastation on organizations continues to climb. Kaspersky Press Release. Online: https://usa.kaspersky.com/about/press-releases/2017_kaspersky-lab-research-shows-ddos-devastation-on-organizations-continues-to-climb

[2] Cimpanu, C. (2021) Microsoft said it mitigated a 2.4 tbps ddos attack. *The Record*. Online: https://therecord.media/microsoft-said-it-mitigated-a-2-4-tbps-ddos-attack-the-largest-ever/

[3] Akamai (2018) Memcached reflection attacks: A new era for DDoS. Online: https://www.akamai.com/uk/en/multimedia/documents/brochure/memcached-reflection-attacks-launch-a-new-era-for-ddos-brochure.pdf

[4] KrebsOnSecurity (2016) Krebsonsecurity hit with record ddos. Blog KrebsOnSecurity. Online: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/

[5] Behal, S.; Kumar, K. (2017) Detection of ddos attacks and flash events using information theory metrics–an empirical investigation. *Computer Communications* 103(C): 18-28. DOI: 10.1016/j.comcom.2017.02.003

[6] Deka, R.; Bhattacharyya, D.; Kalita, J. (2019) Active learning to detect ddos attack using ranked features. *Computer Communications* 145: 203-222. DOI: 10.1016/j.comcom.2019.06.010

[7] Hoque, N.; Kashyap, H.; Bhattacharyya, D. (2017) Real-time ddos attack detection using fpga. *Computer Communications* 110: 48-58. DOI: 10.1016/j.comcom.2017.05.015

[8] Chonka, A.; Singh, J.; Zhou, W. (2009) Chaos theory based detection against network mimicking ddos attacks. *IEEE Communications Letters* 13(9): 717-719. DOI: 10.1109/LCOMM.2009.090615

[9] Nezhad, S.; Nazari, M.; Gharavol, E. (2016) A novel dos and ddos attacks detection algorithm using arima time series model and chaotic system in computer networks. *IEEE Communications Letters* 20(4): 700-703. DOI: 10.1109/LCOMM.2016.2517622

[10] D. Y. e. a. Zhi-jun Wu, Jin Lei (2013) Chaos-based detection of ldos attacks. *Journal of Systems and Software* 86(1): 211-221. DOI: 10.1016/j.jss.2012.07.065

[11] Khan, M.; Ferens, K.; Kinsner, W. (2014) A chaotic measure for cognitive machine classification of distributed denial of service attacks. In: Proc. 2014 IEEE 13th International Conference on Cognitive Informatics and Cognitive Computing, pp. 100–108. DOI: 10.1109/ICCI-CC.2014.6921448

[12] Wu, X. and Chen, Y. (2013) Validation of chaos hypothesis in NADA and improved DDoS detection algorithm. *IEEE Communications Letters* 17(12): 2396-2399. DOI: 10.1109/LCOMM.2013.102913.130932

[13] Chen, Y.; Ma, X.; Wu, X. (2013) DDoS detection algorithm based on preprocessing network traffic predicted method and chaos theory. *IEEE Communications Letters* 17(5): 1052-1054. DOI: 10.1109/LCOMM.2013.031913.130066

[14] Bazeia, D.; Pereira, M.; Brito, A.; Oliveira, B.; Ramos, J. (2017) A novel procedure for the identification of chaos in complex biological systems. *Scientific Reports* 7: a44900. DOI: 10.1038/srep44900

[15] Abbas, R.; Sultan, Z.; Bhatti, S. (2017) Comparative study of load testing tools: Apache JMeter, HP LoadRunner, Microsoft Visual Studio (TFS), Siege. *Sukkur IBA Journal of Computing and Mathematical Sciences* 1(2): 102-108. DOI: 10.30537/sjcms.v1i2.24

[16] FIFA, (2019) 1998 world cup france. FIFA. Online: https://www.fifa.com/worldcup/archive/france1998/matches

[17] CAIDA (2009) Center for applied internet data analysis (caida). Center for Applied Internet Data Analysis (CAIDA),Online: http://www.caida.org/data/passive/ddos-20070804dataset.xml.

[18] Manaf Gharaibeh (2009) Darpa-2009 intrusion detection dataset report. Colorado State University. Online: http://www.darpa2009.netsec.colostate.edu/

[19] Behal, S.; Kumar, K. (2016) Trends in validation of DDoS research. *Procedia Computer Science* 85: 7-15. DOI: 10.1016/j.procs.2016.05.170