Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Everson Luis Rosa Lucion

Perímetro Definido por Software: aumentando os níveis de segurança na autenticação com Single Packet Authorization e Device Fingerprinting

Dissertação de Mestrado

Santa Maria, RS
2018

Everson Luis Rosa Lucion

Perímetro Definido por Software: aumentando os níveis de segurança na autenticação com Single Packet Authorization e Device Fingerprinting

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC), da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Raul Ceretta Nunes

Santa Maria, RS
2018

Lucion, Everson Luis Rosa
Perímetro Definido por Software: aumentando os níveis de segurança na autenticação com Single Packet Authorization e Device Fingerprinting / Everson Luis Rosa Lucion.- 2018.
88 p.; 30 cm

Orientador: Raul Ceretta Nunes
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Informática, RS, 2018

1. Perímetro Definido por Software 2. Autenticação 3. Autorização por um único pacote 4. Fingerprinting de Dispositivo I. Nunes, Raul Ceretta II. Título.

Sistema de geração automática de ficha catalográfica da UFSM. Dados fornecidos pelo autor(a). Sob supervisão da Direção da Divisão de Processos Técnicos da Biblioteca Central. Bibliotecária responsável Paula Schoenfeldt Patta CRB 10/1728.

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

Perímetro Definido por Software: aumentando os níveis de segurança na autenticação com Single Packet Authorization e Device Fingerprinting

elaborada por
Everson Luis Rosa Lucion

como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

Raul Ceretta Nunes, Dr.
(Presidente/Orientador)

Rogério Correa Turchetti, Dr. (UFSM)

Érico Marcelo Hoff do Amaral, Dr. (UNIPAMPA)

Santa Maria, 14 de dezembro de 2018.

DEDICATÓRIAS

- *Gostaria de dedicar este trabalho a Nossa Senhora de Fátima, por ser tão presente e essencial em minha vida;*
- *A todos os meus professores desde o ensino básico até o presente momento, que são de fundamental importância, na construção da minha vida profissional;*
- *Dedico este projeto a minha família e amigos que sempre estiveram presente direta ou indiretamente na minha formação, especialmente, minha esposa Alessandra e minha mãe;*
- *À Direção e colegas do CPD que sempre me incentivaram e apoiaram nas situações em que mais precisei.*

AGRADECIMENTOS

- Ao Prof. Dr. Raul Ceretta Nunes pela sua paciência, conselhos e ensinamentos que foram essenciais e determinantes para o desenvolvimento deste trabalho;
- À UFSM por ser esta Universidade de excelência da qual faço parte como servidor, possibilitando-me o Mestrado de forma gratuita;
- Ao Programa de Pós-Graduação em Ciência da Computação pelo acolhimento e carinho;
- Aos meus colegas do CPD/UFSM que absorveram e mantiveram grande parte das atividades realizadas por mim para cursar o Mestrado;
- Aos meus pais Luiz e Elis, minha esposa Alessandra, pelo apoio incondicional em todas às horas, pelo amor, carinho, incentivo e compreensão.

Não se deve ir atrás de objetivos fáceis, é preciso buscar o que só pode ser alcançado por meio dos maiores esforços.

Albert Einstein

RESUMO

Perímetro Definido por Software: aumentando os níveis de segurança na autenticação com Single Packet Authorization e Device Fingerprinting

AUTOR: Everson Luis Rosa Lucion

ORIENTADOR: Raul Ceretta Nunes

O modelo tradicional de perímetro de rede baseado em *firewall*, permite a comunicação entre os dispositivos antes de efetuarem a autenticação, o que resulta em vulnerabilidades que facilitam diferentes categorias de ataques/invasões. Para mitigar esta vulnerabilidade, a *Cloud Security Alliance* (CSA) propôs o Perímetro Definido por Software (SDP), uma nova abordagem para autenticar antes de a primeira comunicação acontecer. No SDP, o uso de *Single Packet Authorization* (SPA) é fundamental para que o primeiro acesso ocorra apenas após a autenticação do dispositivo. Através da análise do protocolo SDP verificaram-se questões de segurança que precisam ser melhoradas ou abordadas na criação do SPA. Observa-se também que algumas vulnerabilidades ainda persistem, tendo em vistas falhas no modelo TCP/IP quando a identidade de um dispositivo é vinculado ao seu endereço IP. Este trabalho recomenda adequações na arquitetura SDP e definição de um novo padrão de criação e envio do SPA. Ele foi projetado sob aspectos modulares que são incorporados à arquitetura SDP. Além disso, propõem a inclusão na estrutura do SPA de um campo de *fingerprint* de dispositivo, assim como apresenta um método para construir e usar o novo campo com o intuito de solucionar o *gap* temporal entre a autenticação do SPA e conexão para autenticação do usuário. Os resultados demonstram que a solução proposta combate o acesso indevido com o *fingerprinting* de dispositivos e aumenta consideravelmente o grau de dificuldade de detecção, replicação ou leitura dos dados do SPA. Através dos experimentos foi demonstrado que o aumento do tempo de processamento do novo SPA e a geração do *fingerprint* não comprometem a solução e são justificados pelos ganhos nos níveis de proteção.

Pavras-chave: Perímetro Definido por Software. Autenticação. Autorização por um único pacote. *Fingerprinting* de Dispositivo.

ABSTRACT

Software Defined Perimeter: security improvements in authentication with Single Packet Authorization and Device Fingerprinting

AUTHOR: Everson Luis Rosa Lucion

ADVISOR: Raul Ceretta Nunes

The traditional firewall-based network perimeter model enables communication between devices before they authenticate, resulting in vulnerabilities that facilitate different types of attacks/intrusions. To mitigate this vulnerability, the Cloud Security Alliance (CSA) proposed the Software Defined Perimeter (SDP), a new approach to authenticate before the first communication occurs. In SDP, the use of Single Packet Authorization (SPA) is critical for first access to occur only after device authentication. Through the analysis of the SDP protocol there were security issues that need to be improved or addressed in the creation of the SPA. It is also observed that some vulnerabilities still persist, having seen failures in the TCP/IP model when the identity of a device is bound to its IP address. This work recommends adaptations in the SDP architecture and definition of a new pattern of creation and sending of the SPA. It was designed under modular aspects that are incorporated into the SDP architecture. In addition, they propose to include in the SPA structure a device fingerprint field, as well as present a method to construct and use the new field in order to solve the temporal gap between SPA authentication and connection for user authentication. The results demonstrate that the proposed solution fights improper access and considerably increases the degree of difficulty in detecting, replicating or reading SPA data. Through the experiments it has been demonstrated that the increase of the processing time of the new SPA and the generation of the fingerprint do not compromise the solution and are justified by the gains in the levels of protection.

Keyords: Software Defined Perimeter. Authentication. Single Packet Authorization. Device Fingerprinting.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADD	Acrescentar, adicionar
AES	Advanced Encryption Standard
AID	Agente ID
ANSI	American National Standards Institute
CBC	Cipher-Block Chaining
CSA	Cloud Security Alliance
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
HDD	Hard Disk Drive
HMAC	Hash-based Message Authentication Code
HOTP	HMAC-Based One-Time Password
HTTPS	Hiper Text Transfer Protocol Secure
ICMP	Internet Control Message Protocol
ID	Unique Identifier
IP	Internet Protocol
IPv4	IP Version 4
IPv6	IP Version 6
KDF	Key Derivation Funciton
MAC	Message Authentication Code
MER	Modelo Entidade Relacionamento
MFA	Multi-factor Authentication
NAT	Network Address Translation
NIST	National Institute of Standards and Technology
OOB	Out-of-band

OTP	On Time Password
PAC	Proxy Auto-Config
PBKDF2	Password-Based Key Derivation Function 2
PRF	Pseudorandom Function
PRNG	Pseudo-Random Number Generator
RAM	Random Access Memory
RFC	Request for Comments
SDP	Software Defined Perimeter
SHA-1	Secure Hashing Algorithm 1
SHA-2	Secure Hashing Algorithm 2
SMS	Short Message Service
SPA	Single Packet Authorization
SSD	Solid State Drive
SYN	Synchronization
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
TOTP	Time-Based One-Time Password
UDP	User Datagram Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
UUID	Universally Unique Identifier
VPN	Virtual Private Network

SUMÁRIO

1 INTRODUÇÃO

Normalmente, a defesa de perímetro é realizada através de um dispositivo dedicado que controla o acesso à rede privada ao permitir ou negar fluxos de tráfego com base nas diretivas de segurança de uma organização (NARAYANASWAMY, 2008). O modelo de perímetro baseado em *firewall* é comparado a um castelo medieval, cercado com paredes grossas e um único ponto de entrada. Tudo que estiver fora é considerado perigoso e tudo que estiver dentro é confiável. No entanto, a segurança que as barreiras proporcionam no mundo físico não se traduz no ciberespaço. Com a disseminação da computação móvel, o uso de diferentes dispositivos e crescente utilização de serviços baseados em nuvem[1], surgem vetores de ataques adicionais (WARD; BEYER, 2014).

Devido à grande variedade de dispositivos usados em redes de computadores, a segurança cibernética desempenha um papel importante na proteção e melhoria do desempenho das próprias redes ou de sistemas (PUTHAL et al., 2017). Desde os primeiros dias da infraestrutura de TI, as empresas usam a segurança de perímetro tradicional para proteger e bloquear o acesso a recursos internos. O modelo de segurança perimetral baseada em *firewall* funciona bem quando todos os funcionários trabalham, exclusivamente, em construções pertencentes a uma mesma empresa (WARD; BEYER, 2014). Entretanto, o perímetro já não é apenas a localização física da empresa e o que está dentro do perímetro já não é um lugar protegido e seguro para hospedar dispositivos e aplicações pessoais e corporativas. A rede interna é tão repleta de perigo quanto a *Internet* pública (PETERSON; DAVIE, 2013).

As atuais infraestruturas de Tecnologia da Informação estão se tornando mais híbridas[2] e diversificadas que, na atualidade, migram da infraestrutura baseada em *hardware* para a de *software* (PUTHAL et al., 2017). Procurando oferecer aos proprietários de aplicações a capacidade de implantar funcionalidades perimetrais, a *Cloud Security Alliance*[3] (CSA) propôs uma nova ideia para autenticar antes de a primeira comunicação acontecer. Com isso lançou a Especificação 1.0 (BILGER et al., 2014) do protocolo que descreve o Perímetro Definido por Software (do inglês *Software Defined Perimeter* - SDP). A autorização por um único pacote (do inglês *Single Packet Authorization* - SPA) é o passo inicial para ter acesso a um SDP. A sua concepção inclui a ocultação da infraestrutura, de serviços e controles de acesso, bem como a confidencialidade e integridade das comunicações.

O SDP permite a redução de riscos ao diminuir a exposição a ataques cibernéti-

[1] SaaS – *Software as a Service* (*Software* como Serviço), IaaS – *Infrastructure as a Service* (Infraestrutura como Serviço), PaaS – *Platform as a Service* (Plataforma como Serviço).

[2] Estão evoluindo do sistema tradicional destinado a apenas um local para estratégias híbridas que vinculam serviços de TI internos e externos motivados pela existência da nuvem.

[3] Estimula o uso de boas práticas para fornecer garantia de segurança dentro da *Cloud Computing* e prover educação sobre os usos da *Cloud Computing* para ajudar a proteger todas as outras formas de computação. Disponível em: *https://cloudsecurityalliance.org/*

cos. O protocolo surgiu em resposta à urgência de atualizar o estado atual da segurança cibernética, dando início a um novo conceito de arquitetura de segurança (BILGER et al., 2014). A validação de dispositivo mitiga o roubo de credenciais e os ataques resultantes de *impersonation*[4]. A validação do dispositivo está além do escopo desta versão 1.0 do SDP, mas deve ser abordada em versões futuras.

Para fornecer um nível básico de segurança, o Perímetro Definido por Software começa com disponibilidade e visibilidade zero. A Especificação 1.0 (BILGER et al., 2014) recomenda um modelo de segurança para verificar a identidade de dispositivos e usuários antes de conceder acesso a sistemas ou redes. Os recursos principais do SDP devem ter a capacidade em autenticar e validar dispositivos, autenticar e autorizar usuários, garantir comunicações criptografadas bidirecionais e provisionar serviços dinamicamente (BILGER et al., 2014).

Como o SDP adota uma técnica para autenticar primeiro e depois se comunicar, os atacantes possuem dificuldades para avaliar propriedades de segurança sem serem autenticados primeiro. Neste sentido, verificaram-se questões de segurança que precisam ser melhoradas e abordadas durante o processo de autenticação para ingresso em um SDP.

1.1 Definição do Problema de Pesquisa e Justificativas

Através da análise da Especificação 1.0 (BILGER et al., 2014) se verificou questões de segurança do SPA que precisam ser melhoradas ou abordadas durante o processo de sua criação e envio: 1) por sugerir o TCP, o princípio de pacote único não foi previsto; 2) os campos do *payload* são transportados sem criptografia; 3) o uso da função HOTP (VIEW et al., 2005) para formação da senha única a torna válida por tempo indeterminado; 4) questões relacionadas ao uso do IPv6 não foram tratadas; 5) a senha secreta do usuário usada não oferece proteção contra ataques de força bruta ou dicionário; 6) a verificação de integridade dos campos do *payload* não foi prevista; 7) a autenticidade de dados deveria ser prevista e aplicada a todo *payload*; 8) ações que atendam a confidencialidade de tráfego não foram previstas. Ainda, *a* necessidade de ocultação da conexão TCP subsequente é um problema que persiste em técnicas de autenticação via SPA.

Outro problema ocorre quando a identidade é vinculada a um endereço IP, pois é assumido que apenas um dispositivo acessará um servidor de um determinado endereço. A tradução de endereços de rede (do inglês *Network Address Translation* - NAT), *proxy* e técnicas similares têm um mesmo IP compartilhado centenas ou milhares de vezes, bem como ataques de IP *spoofing* podem ocorrer. Portanto, nesse

[4] Ataque no qual um adversário assume com sucesso a identidade de uma das partes legítimas no sistema ou em um protocolo de comunicação.

cenário, não há certeza de que o endereço IP, que deseja acesso ao perímetro, seja aquele dispositivo que teve autenticação validada e sua autorização permitida via SPA.

Em síntese, um novo modelo de perímetro é definido pela *Cloud Security Alliance*, porém há questionamentos ainda sem respostas. A Especificação 1.0 possui aspectos que podem ser melhorados? O SPA possui forte proteção contra ataques? Qual a melhor forma de solucionar o problema da conexão TCP subsequente? A validação do dispositivo deve fazer parte do processo de autenticação para evitar ataques de roubo de credenciais? Vincular a identidade de um dispositivo ao seu IP é seguro?

Pelo exposto, verifica-se que reforçar a segurança perimetral é um desafio e que a Especificação v1.0 do SDP auxilia, mas não mitiga vulnerabilidades importantes e que devem ser tratadas.

1.2 Contribuições

Este trabalho propõe adequações na arquitetura SDP descrita pela Especificação 1.0 (BILGER et al., 2014) e uma nova abordagem projetada sob aspectos modulares de criação do SPA. Os módulos de originalidade e autenticidade foram fortalecidos, pois já faziam parte da Especificação 1.0 do SDP. Seis novos módulos foram criados e incorporados ao modelo: pontualidade, integridade e confidencialidade de dados, segurança da senha do usuário, compatibilidade com o protocolo IPv6 e confidencialidade de tráfego. Desenvolveu-se ainda uma forma segura de estabelecer a ocultação da conexão TCP subsequente para autenticação inicial de usuários, tornando-a mais robusta.

O trabalho propõe também uma extensão à arquitetura SDP para resolver falhas que ainda persistem no modelo TCP/IP com a possibilidade de exploração do *gap* temporal (tempo em que o controlador de perímetro fica aguardando pela conexão TCP/443 do IP informado via SPA para iniciar a autenticação do usuário). Nesse tempo, ataques (e.g., *spoofing)* podem ocorrer entre o final do SPA e a conexão TLS[5] subsequente. Um novo campo de *fingerprint* (i.e., resultado da construção da assinatura digital de um dispositivo computacional) é adicionado na estrutura de *payload* do SPA. Um método de manipulação (criação e uso) do novo campo é apresentado. Usando atributos únicos e derivação de chaves, o método obtém um *fingerprint* robusto que dificulta falsificações. Neste sentido é possível verificar se a conexão TLS do SDP, que sucede a autenticação do SPA, está sendo realizada pelo mesmo usuário e dispositivo que de fato enviou o SPA.

[5] TLS (Segurança da Camada de Transporte, em inglês *Transport Layer Security*).

1.3 Principais Resultados

Os resultados experimentais mostram que o aumento do tempo de processamento de um SPA de 0,26 para 9,37 ms no Cliente e de 0,49 para 13,44 ms no *Controller back end* se justifica pelos ganhos relevantes dos níveis de proteção. A validação de segurança demonstra o fortalecimento e aumento significativo da proteção contra ataques. Nesse sentido, a adição dos módulos propostos ao SDP mostra-se viável. O transporte do campo *timestamp* e *counter* dentro do *payload* SPA reduz até 90 vezes o tempo de validação de senhas. A derivação de chaves e a junção dos protocolos HOTP e TOTP fortalecem a segurança das senhas. A definição de uma nova arquitetura para envio do SPA e estabelecimento do TLS mútuo propiciam a ocultação, escalabilidade e redundância desejadas.

O resultado do *fingerprinting* de dispositivo mostra que a solução tem um baixo impacto no desempenho e no processo de autenticação. Com a solução são mitigados ataques via alteração do modelo de segurança de perímetro definido por *software*. A adoção de *fingerprint* do dispositivo passa a atuar como fator principal na identificação do Cliente, onde o endereço IP é colocado em segundo plano.

1.4 Organização de Texto

Este trabalho está organizado da seguinte forma: o capítulo 2 apresenta os conceitos necessários para a compreensão do trabalho. O capítulo 3 relata a metodologia de trabalho e a solução proposta. O capítulo 4 descreve as proposições de alteração para implantação de um Perímetro Definido por Software com melhorias na segurança do *Single Packet Authorization* e autenticação de usuários. O capítulo 5 apresenta uma solução para aumento dos níveis de segurança na autenticação no SDP com *Device Fingerprinting*. O capítulo 6 detalha a validação de segurança das soluções propostas. Por fim, o capítulo 7 traz as conclusões finais e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O perímetro pode ser caracterizado como uma linha imaginária que separa uma empresa (seus serviços, computadores, servidores, redes, etc.) de outras redes, como a *Internet*. Por conta disso, a segurança de perímetro continua sendo um elo primário e extremamente importante para qualquer organização que deseja estar mais segura com o uso da *Internet*.

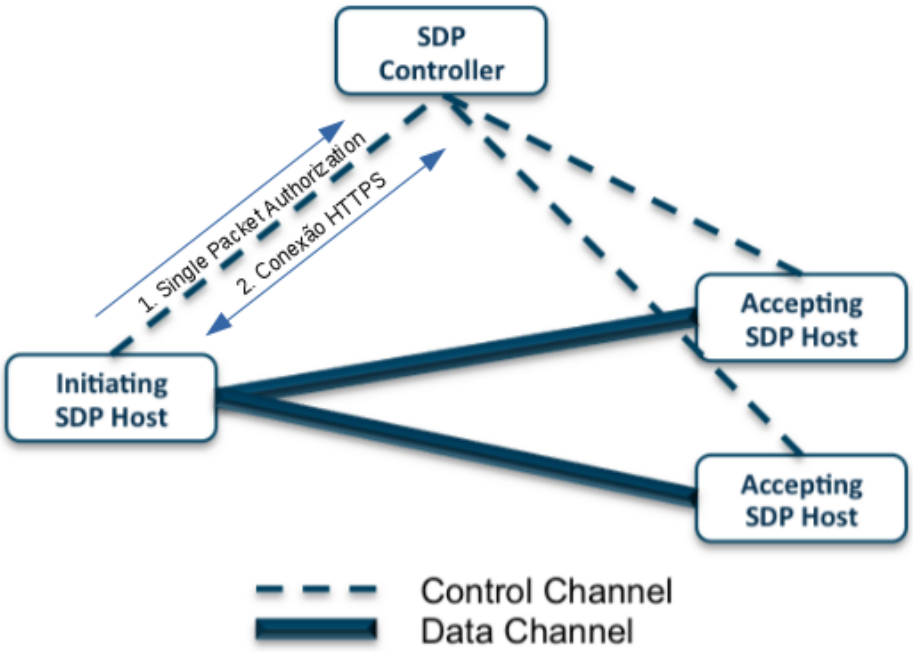
Para a compreensão deste trabalho, são apresentados conceitos importantes a seguir. A seção 2.1 aborda e descreve o Perímetro Definido por Software, a seção 2.2 apresenta a definição de *Single Packet Authorization,* a seção 2.3 descreve as Funções de *Hash* Criptográfico, na seção 2.4 destaca-se a Função de Derivação de Chaves, a seção 2.5 apresenta o conceito de *On Time Password*, a seção 2.6 define a identificação de dispositivos através de *Device Fingerprinting*, a seção 2.7 conceitua Identificação, Autenticação e Autorização, a seção 2.8 evidencia a autenticação via canais *Out-of-band,* seção 2.9 expõe a Autenticação Multi-factor e por fim, a seção 2.10 exemplifica as maneiras pelas quais seres humanos ou dispositivos podem serem autenticados através dos fatores de autenticação.

2.1 O Perímetro Definido por Software

Atualmente, o perímetro não é apenas a localização física da empresa e o que está dentro já não é um lugar protegido e seguro para hospedar dispositivos e aplicações pessoais e corporativas. O Perímetro Definido por Software é uma arquitetura de segurança desenvolvida por membros da *Cloud Security Alliance*. O documento inclui a ocultação da infraestrutura, de serviços, de aplicações, dos controles de acesso bem como a confidencialidade e integridade das comunicações (BILGER et al., 2014). No SDP, o plano de controle é separado do plano de dados para permitir um sistema completamente escalável, conforme ilustra a Figura 1.

Na arquitetura SDP, o esquema de controle é definido pelo Cliente ou *host* de iniciação (*Initiating Host* - IH) e pelo *host* de aceitação (*Accepting Host* - AH). Nesse plano, ambos se comunicam com o *Controller*. Já o plano de dados descreve a forma de comunicação entre o IH e o AH. Além disso, todos os componentes devem ser projetados para fins de escalabilidade ou redundância (*uptime*).

Figura 1 – Arquitetura SDP: 1) envio de um SPA do Cliente ao *Controller*. 2) estabelecimento da conexão TLS subsequente.

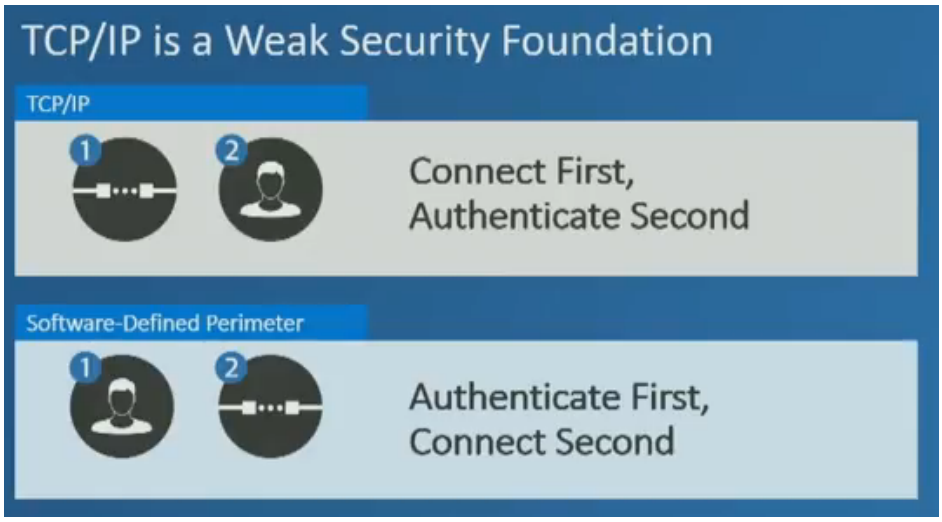


Fonte: Cloud Security Alliance.

A ideia chave do SDP é eliminar a fraqueza do protocolo TCP/IP no que diz respeito ao estabelecimento da conexão. O modelo de segurança baseado em TCP/IP fornece uma porta aberta para os invasores, que podem se comunicar antes da autenticação, o que deixa uma falha de segurança para que um atacante entre no sistema antes de comprovar suas credenciais (PUTHAL et al., 2017).

O TCP permite a comunicação entre os dispositivos antes de efetuarem a autenticação, permitindo aos invasores entrar no processo de transmissão de dados (PUTHAL et al., 2017). No SDP a autenticação do dispositivo deve ser realizada antes da conexão, via SPA. A Figura 2 mostra a diferença entre a segurança tradicional baseada em TCP/IP e o SDP.

Figura 2 – Diferença entre a segurança tradicional e o Perímetro Definido por Software.

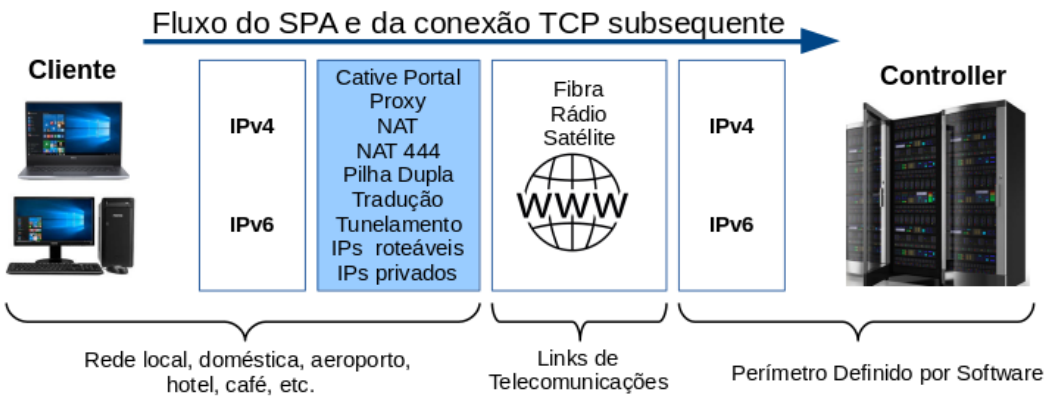


Fonte: Cloud Security Alliance.

O uso de *firewall* na arquitetura SDP lembra o perímetro tradicional em redes. Mas o *firewall* é usado em outro contexto, pois mesmo o SDP dando a ideia de buraco negro (no contexto de redes de computadores, é um lugar onde os pacotes enviados para um determinado destino são todos descartados), o uso do *firewall* se faz necessário como um mecanismo de gerência e segurança.

A Figura 3 apresenta uma visão das possíveis combinações entre arquiteturas e protocolos que uma conexão TCP/IP pode ter entre um Cliente e um *Controller*. Essa visão é necessária para compreender quais desafios ou problemas ainda não foram resolvidos ou abordados no SDP. Observa-se que a origem das conexões são diversas, abrangendo desde a rede local, até redes domésticas, aeroportos, hotéis, cafés, etc.

Figura 3 – Arquitetura conceitual em alto nível de possíveis conexões TCP/IP entre um Cliente e um Perímetro Definido por Software.



Fonte: o Autor.

As conexões TCP/IP que se originam redes IPv4 e IPv6, muitas vezes, precisam sofrer algum tipo de alteração do IP de origem antes de percorrerem os *links*

de telecomunicações até um Perímetro Definido por Software. Quando um dispositivo está vinculado ao seu endereço IP, presume-se que apenas ele acessará um recurso ou aplicação. Essa suposição é quebrada, principalmente, com ataques de IP *spoofing* ou utilização de NAT, *proxy* e mecanismos similares.

2.2 *Single Packet Authorization*

Segundo (BILGER et al., 2014) o SPA é usado para inicializar a comunicação nos seguintes casos: IH-Controller, AH-Controller, e IH-AH.

Desde 2005, o SPA vem sendo usado para proteger as comunicações baseadas em IP. O conceito foi amplamente estendido e aprimorado desde então por *Michael Rash* através do projeto *fwknop* (RASH, 2016). O SPA requer apenas um único pacote criptografado para comunicar várias informações, incluindo o acesso desejado através de uma política de *firewall* e/ou enviar comandos para executar no sistema de destino.

Geralmente, em um mecanismo SPA, são usados os protocolos UDP, TCP ou ICMP. O pacote único é enviado a uma porta destino específica. O servidor detecta o pacote, valida os dados incluídos dentro do *payload* e solicita ao serviço de *firewall* incluir o IP e porta informados em suas regras de permissão. A regra IP/porta estará ativa por um período pré-definido entre as partes, geralmente de 30 segundos. Logo após, o dispositivo Cliente pode conectar-se ao servidor (ZORKTA; ALMUTLAQ, 2012).

A Especificação 1.0 (BILGER et al., 2014) relata que o SPA fornece os seguintes benefícios de segurança para os servidores protegidos por ele :

- Esconde o servidor: o servidor não responderá a nenhuma conexão de nenhum Cliente até que tenha sido fornecido um autêntico SPA;
- Atenua ataques de negação de serviço no TLS: servidores voltados para a *Internet* que executam o protocolo *https* são altamente suscetíveis a ataques de negação de serviço (DoS). O SPA atenua esses ataques porque permite ao servidor descartar a tentativa de TLS DoS antes de entrar no *handshake* TLS;
- Detecção de ataque: o primeiro pacote para um AH de qualquer outro *host* deve ser um SPA. Se um AH receber outro pacote, deve ser visto como um ataque. Portanto, o SPA permite que o SDP determine um ataque baseado em um único pacote malicioso.

2.3 Funções de Hash Criptográfico

Uma função de *hash* criptográfico, comumente conhecida como *hash* – é um algoritmo matemático que transforma qualquer bloco de dados de comprimento variável em uma série de caracteres com comprimento fixo. Independentemente do

comprimento dos dados de entrada, terá sempre uma informação de saída de mesmo comprimento. A família mais bem conhecida de funções de *hash* são a MD-SHA (MD5, SHA-1 e SHA-2).

O SHA-2 foi finalizado em 2009 pelo NIST (*National Institute of Standards and Technology*), padrão que especifica algorítimos de *hash* como o SHA-1, SHA-224, SHA-256, SHA-384 e SHA-512 (Information Technology Laboratory, 2015).

Como função de *hash*, o *Security Hash Algorithm-256* (SHA-256) pode ser aplicado em muitos aspectos, como uma assinatura digital, verificar a integridade de dados e validação da identidade. SHA-256 gera uma saída de 256 *bits* ou 32 *bytes* em um processo teoricamente não reversível (BAI; LI, 2009). O SHA-256 tem muitas vantagens como capacidade anti-ataque, fácil implementação e boa portabilidade. Então, o SHA-256 é a base e o núcleo de muitos métodos populares de criptografia.

Um uso muito comum da função SHA-256 é ser usada em um processo de autenticação de mensagens usando funções *hash* criptográficas e uma chave secreta compartilhada entre o servidor e o Cliente, recurso chamado de HMAC (*Hash-based Message Authentication Code*) (LEE; LIM; LEE, 2010).

2.4 Função de Derivação de Chaves

Uma Função de Derivação de Chaves (do inglês *Key Derivation Funciton* - KDF) é usada para gerar uma ou mais chaves a partir de uma chave mestra (senha secreta). A derivação de chaves é usada para garantir a segurança da informação e proteger dados eletrônicos, quando eles estão guardados ou sendo transmitidos em canais inseguros. O principal objetivo de segurança é que as chaves derivadas geradas pela KDF sejam indistinguíveis, mesmo quando as entradas públicas são fornecidas ao adversário (CHUAH; DAWSON; SIMPSON, 2013).

A *Password-Based Key Derivation Function* 2 ou função PBKDF2 - RFC 2898 (KALISKI, 2000) é usada para derivar chaves de suficiente tamanho.

A PBKDF2 usa a HMAC-SHA-256 - RFC 6234 (HANSEN; 3RD, 2011) como *Pseudorandom Function* (PRF). A PBKDF2 pode ser implementada através da biblioteca *Openssl*[6]. A função HMAC-SHA-256 gera uma saída de 32 *bytes* oferecendo mais proteção contra ataques de força bruta ou dicionário. Abaixo, descreve-se a função de derivação de chaves PBKDF2:

$$DK = PBKDF2(HmacSha256, Password, Salt, c, dkLen)$$

- DK: chave derivada com comprimento de *dkLen* octetos;
- PBKDF2: algorítimo proposto pela RFC 2898;
- HMAC-SHA-256: *Pseudorandom Function* usada;

[6] Disponível em: https://www.openssl.org/docs/man1.1.0/crypto/PKCS5_PBKDF2_HMAC_SHA1.html

- *Password*: chave mestra ou senha do usuário;
- *Salt:* caracteres incorporados ao *Password* e simétricos entre as aplicações (Cliente e C*ontroller)*, conhecido como salto criptográfico e de conhecimento público. Ter um *salt* adicionado à senha reduz a capacidade de usar *hashes* pré-computados (*rainbow tables*) para ataques;
- c: número de iterações desejado;
- *dklen*: tamanho desejado da chave derivada em *bytes.*

A ideia por trás do PBKDF2 e de qualquer algorítimo de *password hashing*, é de que ele seja propositalmente lento. A intenção é elevar custos de ataques de força bruta ou dicionário, o que inevitavelmente aumenta o custo da autenticação do usuário legítimo. O número originalmente recomendado de 1000 iterações para o PBKDF2, por exemplo, tinha como objetivo que o processo demorasse cerca de 1 segundo. Em processadores modernos, o número de iterações precisa ser aumentado para que se chegue a patamares similares (ou, pelo menos, o tempo de processamento precisa ser claramente definido na solução, já que ele é configurável). Por ser um valor configurável, cada administrador de sistemas deverá definir o tempo de processamento que irá impactar na autenticação do usuário, tendo em vista aumento na proteção contra ataques de força bruta ou dicionário.

2.5 *On Time Password*

Um atacante poderia capturar e retransmitir uma cópia de um SPA. Nesse sentido, os sistemas de computação poderiam ser alvos de ataques de repetição. Uma solução possível seria a utilização de senhas de uso único (e.g., *One Time Password* - OTP) (ZHAO, 2013). Na Especificação 1.0 (BILGER et al., 2014) se utiliza a função HOTP: *HMAC-Based One-Time Password* - RFC 4226 (VIEW et al., 2005) que fornece leve grau de originalidade. Essa função é baseada em um contador mantido sincronizado entre o *Controller* e o Cliente. Não há orientação para manter essa variável secreta durante o envio do SPA.

O protocolo HOTP não impediria que o atacante interceptasse o SPA por um tempo e depois o transmitisse. Além disso, caso uma senha HOTP seja comprometida, será potencialmente válida por um tempo indeterminado. O protocolo TOTP: *Time-Based One-Time Password* - RFC 6238 (VIEW et al., 2011) normatiza o uso de senhas baseado no tempo humano e usa o relógio do sistema para criar e validar OTP. Cada implementação pode definir a tolerância de validade da senha ou considerar uma leve dessincronização dos relógios (BEIKVERDI; TAN, 2012).

2.6 Device *Fingerprinting*

Fingerprint é definido formalmente como um conjunto de itens de informação, que caracteriza um dispositivo ou aplicação o qual pode ser representado através de um número ou *hash*. Já ***fingerprinting*** é estabelecido como um processo do qual um aplicativo ou observador identifica de maneira única um dispositivo ou uma instância de aplicação com base em um conjunto de diferentes informações (COOPER et al., 2013).

A identificação do dispositivo é de grande importância para autenticação segura de usuários. Cada dispositivo tem muitas características, como sistema operacional, *hardware, softwares* e características do navegador. A ideia geral é extrair informações sobre os *softwares* e seus estados, o sistema operacional e suas atualizações e, também, a identificação dos componentes de *hardware* (SHEN et al., 2017; GAO; CORBETT; BEYAH, 2010). Essas informações são combinadas em uma sequência de caracteres e uma identificação de *fingerprint* é criada (normalmente usando uma função de *hashing)*. A estabilidade do *fingerprint* obtido (às vezes chamada de "ID de *hash*") dependerá dos atributos associados ao computador do usuário (e.g., informações do sistema operacional, fontes ou versão de aplicativos, etc.) (LUANGMANEEROTE; ZALUSKA; CARR, 2017).

2.7 Identificação, Autenticação e Autorização

A *Internet* consiste em trilhões de redes públicas, privadas, governamentais e não governamentais de âmbito local a global, ligadas por uma ampla gama de tecnologias de rede. Nesse sentido, a necessidade da segurança e privacidade se tornou uma questão importante.

Três aspectos iniciais relacionados à segurança e privacidade envolvem a identificação, a autenticação e a autorização. Esses três conceitos inter-relacionados representam o núcleo de um sistema de segurança da informação. A **identificação** é o processo que ajuda a reconhecer uma entidade reivindicada, que pode ser uma pessoa, uma máquina ou outro recurso, como um *software*. A **autenticação** é o processo através do qual um requerente prova quem afirma ser (RAHMAN; SHUVA; ALI, 2016). Como mencionado, a identificação e a autenticação são regras para identificar uma requisição de uma entidade. Já no processo de **autorização**, as regras são usadas para decidir qual o nível de acesso é permitido a uma entidade autenticada para um particular recurso de sistema (DOULIGERIS; SERPANOS, 2007).

No entanto, segundo Forouzan (2007) existem duas diferenças entre autenticação de mensagem e autenticação de entidade. A primeira pode não acontecer em um tempo real, a autenticação de entidade deve ser em tempo real. A entidade de quem a identidade precisa ser provada é chamada de **requerente**; a parte que tenta provar a identidade do requerente é chamada de **verificador.** Em uma autenticação de

mensagem, o requerente pode estar ou não presente no processo de comunicação. No entanto, na autenticação de entidade o requerente precisa estar *online* e fazer parte do processo. Isso quer dizer que quando há uma autenticação de entidade, não é enviada ou recebida nenhuma mensagem real de comunicação envolvida, até que o requerente seja autenticado pelo verificador. Na autenticação de entidade, o reclamante precisa se identificar para o verificador.

2.8 Autenticação via Canais *Out-of-band*

Autenticação via canais *Out-of-band* (OOB) refere-se a processos de autenticação em que seus métodos são transmitidos através de diferentes canais. Em redes de computadores, é um fluxo de dados separado do fluxo principal. Quando os fatores de autenticação são transmitidos por um único dispositivo/canal, um usuário mal-intencionado ao estabelecer o controle do canal/dispositivo, pode capturar os fatores de autenticação .

A transmissão de uma senha única (OTP) para um *smartphone* tem sido tradicionalmente considerada um método *out-of-band*. No entanto, se o mesmo telefone for usado para enviar o OTP - por exemplo, via navegador *web* - a eficácia do OTP, como um fator secundário, é efetivamente anulada. O transporte *out-of-band* de mecanismos de autenticação é um controle adicional que pode aumentar o nível de garantia de autenticação de vários fatores. O processo de autenticação deve estabelecer controles para garantir que o indivíduo que está tentando a autenticação é, de fato, o usuário legítimo de posse do fator de autenticação (COUNCIL, 2017).

2.9 Autenticação Multi-factor

Entre muitas abordagens de verificação, os sistemas geralmente dependem de senhas estáticas para autenticação de usuários. A proliferação do roubo de identidade reforça que a utilização de métodos de fatores únicos de autenticação (e.g., *user e password*) não são mais eficientes para o controle da segurança. Nesse sentido, é recomendado investir em estratégias que proporcionem maior segurança nos processos de identificação e autenticação de dispositivos e usuários. A intenção da autenticação de múltiplos fatores (do inglês *Multi-Factor Authentication* - MFA) é fornecer um grau mais alto de garantia da identidade do indivíduo tentando acessar um recurso – como um local físico, dispositivo de computação, rede ou serviço – criando um mecanismo de camadas múltiplas em que um usuário não autorizado teria que ultrapassar de modo a obter acesso (RAHMAN; SHUVA; ALI, 2016).

O mundo está se movendo para um novo modelo que dispensa uma rede corporativa privilegiada e o acesso não depende da localização da rede de um usuário. Todo o acesso aos recursos da empresa é totalmente autenticado, autorizado e criptografado

com base no estado do dispositivo e nas credenciais dos usuários (WARD; BEYER, 2014) .

Como abordado na seção 2.1, a *Cloud Security Alliance* lançou em 2014 a especificação do protocolo que descreve o Perímetro Definido por Software. O objetivo é prover conexão segura para servidores e se proteger de ataques cibernéticos. O ingresso no Perímetro Definido por Software é restrito a um agente intermediário de confiança. Ele permite o acesso seguro e confiável aos serviços e conjuntos de dados através da autenticação de usuários e seus dispositivos de conexão (BILGER et al., 2014).

A autenticação é a ação inicial para obter acesso à infraestrutura SDP, por isso a análise de riscos identifica pontos críticos durante o processo de autenticação do dispositivo e do usuário. A análise de riscos determinará quais mecanismos de autenticação deveriam ser usados no processo de validação das credenciais para acesso ao Perímetro Definido por Software. A autorização baseada em riscos envolve a coleta de tanta informação quanto possível sobre o contexto do evento de autenticação (ou “cerimônia”) para depois avaliar a probabilidade de que este é o usuário correto, e atribui um fator de risco para julgar a quantidade de autorização que usuário deve ter (KEARNS, 2009).

Se algo é muito seguro, o acesso (realizar *login*) pode se tornar muito difícil, ninguém vai querer usá-lo. É por isso que as pessoas reutilizam senha e *login* mais simples em dispositivos ou acessos a serviços na *Internet*. Se é muito fácil de usar, mas não seguro, é inerentemente arriscado. A melhor maneira de cumprir esses dois propósitos é fazer com que o segundo ou terceiro fator de autenticação sejam fáceis e seguros.

Confiar eu um único fator pode ser inseguro. O ideal é combinar pelo menos dois fatores para possuir uma autenticação confiável. A autenticação multifatorial é geralmente mais segura do que a de fator único.

Neste sentido, é necessário garantir que a pessoa que está do outro lado tentando acessar uma determinada informação, seja a pessoa que realmente tenha a autorização necessária e sua veracidade precisa ser comprovada. Para isso, um ser humano ou dispositivo pode ser autenticado em um sistema computacional através de vários fatores de autenticação.

2.10 Fatores de Autenticação

Existem maneiras pela qual um ser humano ou outro dispositivo pode ser autenticado em um sistema computacional, esses modos são chamados de fatores de autenticação (RAHMAN; SHUVA; ALI, 2016). Cada fator de autenticação abrange uma variedade de opções para autenticar o usuário ou dispositivo antes de prover acesso

aos sistemas requisitados pelo mesmo. Uma estrutura bem aceita para autenticação é “algo que você conhece” emparelhado com “algo que você tem” (GROSSE; UPADHYAY, 2013).

Os sistemas de autenticação geralmente são categorizados pelo número de fatores que eles incorporam. Ao lidar com diferentes situações, o contexto é o elemento-chave utilizado para inferir possíveis ações e necessidades de informações. Alguns dos fatores de autenticação amplamente reconhecidos pelos seres humanos são os seguintes (RAHMAN; SHUVA; ALI, 2016; HASAN; KHAN, 2017; BORTEN, 2017; SOUZA; BURLAMAQUI; SOUZA FILHO, 2017):

- **Conhecimento**: algum conhecimento que o usuário possui (uma pergunta de segurança, chave secreta, frase-senha, um padrão, chave privada, senha ou PIN *number*[7], etc.);
- **Possessão**: o que o usuário tem (um *token* físico em que ele pode facilmente levar consigo, tal como um *smartphone*, cartão de identificação, cartão inteligente, cartão de crédito, dispositivo *usb*, uma conta de *e-mail* ou outro dispositivo para gerar um código temporário de uso único, etc.);
- **Característica física**: o que ele é, informações que vêm de partes do corpo humano ou chamadas de biométricas (e.g., íris, padrão retinal, reconhecimento facial, impressões digitais, reconhecimento de voz, padrão da palma da mão, etc.);
- **Ação**: é o que o usuário pode fazer, uma atividade (assinatura, gesto, etc.);
- **Interações anteriores**: a autenticação pode ser baseada em um ou mais eventos passados de um usuário interagindo com um sistema, que nos permite postular a autenticação e o controle de acesso que podem sofrer a influência de interações anteriores do usuário com o sistema de autenticação;
- **Localização**: o lugar onde o usuário está (um ponto geográfico, zona geográfica, etc.).

[7] Do inglês PIN, acrónimo de *personal identification number*, «número de identificação pessoal» usado como senha de acesso a um sistema eletrônico bancário, telefônico, etc.

3 SOLUÇÃO PARA AUTENTICAÇÃO NO SDP

O SDP é um modelo moderno de controle de perímetro. As especificações da CSA orientam soluções básicas, mas soluções sólidas demandam investigações. Este trabalho desenvolve soluções robustas para autenticação no SDP que seguem os seguintes princípios:

1) Comunicação inicializada entre o *host* de inicialização (Cliente) e o *Controller*. As outras comunicações previstas pelo protocolo SDP seguem suas características originais;

2) A identificação da entidade reivindicada (requerente) é composta por um dispositivo (físico ou lógico), um sistema operacional e uma pessoa;

3) Autenticação corresponde a de um requerente (dispositivo e pessoa);

4) Não são abordados aspectos relacionados a autorização;

5) Quem comprova a identidade do requerente é o *Controller,* comumente nomeado de verificador;

6) O processo de autenticação do usuário e do dispositivo segue o modelo multifactor baseado em riscos (moderado e alto);

7) Um novo acesso é caracterizado como risco “alto”. Nos demais acessos o risco é qualificado como “moderado”;

8) A cada alteração dos parâmetros associados aos dados do usuário, *hardware* ou sistema operacional, o fator de risco torna-se “alto”;

9) Um SPA pode incluir *fingerprint*.

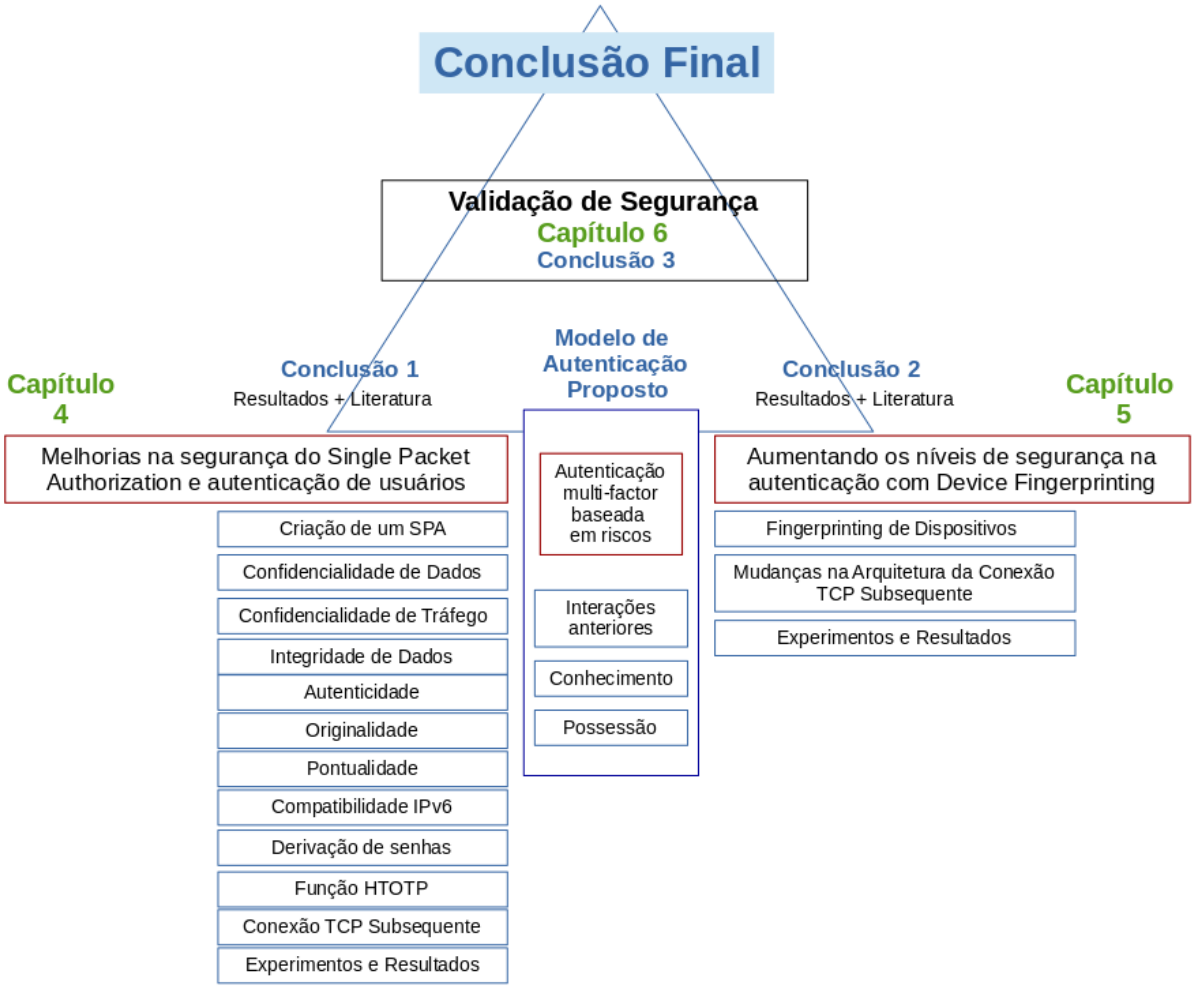
A seguir é apresentado a metodologia usada para desenvolver a solução de autenticação proposta.

3.1 Metodologia do Trabalho

Esta seção apresenta a metodologia proposta para realizar o processo de criação do novo SPA e da nova arquitetura SDP, além do processo de *fingerprinting* de dispositivos. O trabalho analisou o que está em conformidade com o modelo e propôs melhorias, novos conceitos e técnicas que podem ser incorporadas, com foco nos mecanismos de autenticação de usuários e dispositivos.

A Figura 4 apresenta um organograma de como foi conduzida a pesquisa, seus passos e etapas.

Figura 4 – Organograma desta pesquisa com suas relações e dependências.



Fonte: o Autor.

Em sua base e ao centro, o estudo possui um modelo de autenticação *multi-factor* que é incorporado às implementações de criação do SPA e *fingerprinting* de dispositivos. Também, na base, as duas conclusões parciais dos Capítulos 4 e 5 formam o embasamento científico desse trabalho. Ao centro, a validação de segurança e sua conclusão, e ao topo a conclusão.

O capítulo 4 propõe adequações e melhorias na arquitetura SDP através de uma nova abordagem modular de criação do SPA. Desenvolveu-se, também, uma forma de ocultar da conexão TCP subsequente (TLS mútuo) para autenticação inicial de usuários.

Já o capítulo 5 apresenta o *fingerprinting* de dispositivos como uma extensão à arquitetura SDP para resolver falhas que ainda persistem no modelo TCP/IP com a possibilidade de exploração do *gap* temporal (tempo em que o controlador de perímetro fica aguardando pela conexão TCP/443 do IP informado via SPA para iniciar a autenticação do usuário).

No capítulo 6 é realizada a validação do trabalho, ela é baseada em análise

de segurança voltada a ataques e como eles são mitigados. Neste sentido, é apresentada uma descrição de 20 ataques e uma explanação de como são resolvidos, levando-se em conta os recursos que foram descritos e implementados. O capítulo 6 apresenta a avaliação de segurança deste trabalho.

Assim, a solução proposta e implementada por este trabalho:

- Define um novo modelo de criação de um *Single Packet Authorization*;
- Gera o *fingerprint* e valida dispositivos para ingresso no perímetro;
- Cria uma arquitetura para recebimento do SPA e estabelecimento da conexão TCP subsequente e;
- Propõe diferentes fatores e métodos de autenticação;
- A partir de ataques selecionados de publicações prévias que envolvam defesa de perímetro, realiza a validação da segurança, tendo em vista os recursos descritos e implementados.

Como resultados, este trabalho oferece melhorias no modelo SDP para validar e autenticar dispositivos e autenticar usuários para acesso inicial entre o Cliente e o Controlador do perímetro.

4 MELHORIAS NA SEGURANÇA DO *SINGLE PACKET AUTHORIZATION* E AUTENTICAÇÃO DE USUÁRIOS

Através da análise da Especificação 1.0 (BILGER et al., 2014) se verificaram questões de segurança do SPA que precisam ser melhoradas ou abordadas durante os processos de sua criação e envio: por sugerir o TCP, o princípio de pacote único não foi previsto; os campos do *payload* são transportados sem criptografia; o uso da função HOTP (VIEW et al., 2005) para formação da senha única a torna válida por tempo indeterminado; questões relacionadas ao uso do IPv6 não foram tratadas; a senha secreta do usuário não oferece proteção contra ataques de força bruta ou dicionário; a verificação de integridade dos campos do *payload* não foi prevista; a autenticidade de dados deveria ser prevista e aplicada a todo *payload*; ações que atendam a confidencialidade de tráfego não foram previstas. E por fim, o *gap* da conexão TCP subsequente é um problema, que ainda persiste em técnicas de autenticação via SPA.

Este capítulo propõe adequações e melhorias na arquitetura SDP apresentada pela Especificação 1.0 (BILGER et al., 2014) e uma nova abordagem projetada sob aspectos modulares de criação do SPA. Os módulos de originalidade e autenticidade foram fortalecidos, pois já fazem parte do protocolo. Seis novos módulos foram criados: pontualidade, integridade e confidencialidade de dados, derivação de senhas, compatibilidade com o protocolo IPv6 e confidencialidade de tráfego. Desenvolveu-se, também, uma forma segura de estabelecer a ocultação da conexão TCP subsequente para autenticação inicial de usuários, tornando-a mais robusta.

Este capítulo está organizado da seguinte forma. Os trabalhos relacionados são discutidos na seção 4.1. Na sequência, a seção 4.2 apresenta o projeto do novo SPA. Posteriormente, a seção 4.3 expõe a criação modular do SPA. A seção 4.4 relata a conexão TCP subsequente. A seção 4.5 apresenta os experimentos e resultados. Por fim, a seção 4.6 evidencia a discussão e a seção 4.7 traz a conclusão parcial.

4.1 Trabalhos Relacionados

Em (TARIQ; BAIG; SAEED, 2008) os autores relatam que a falta de associação entre o processo de autenticação e a conexão TCP subsequente é um problema que ainda persiste em ambas as técnicas de autorização passiva (SPA e *port knocking*[8]). Esse problema permite que um invasor se conecte a um servidor protegido em nome de um cliente válido, após o cliente ter autenticado com sucesso no *firewall*, mas antes de estabelecer uma conexão TCP com o servidor. Os autores criaram uma solução que compartilha um *nonce*[9] criptografado entre o cliente e o *firewall* durante o envio do SPA.

[8] Tentativa de conexão em uma sequência de portas predeterminada.

[9] Palavra de uso único (n = *Number* e *once* = Uma vez, em inglês), o N pode usar letras também.

O *firewall* somente aceitaria tentativas de conexões, caso o primeiro TCP SYN tivesse o *nonce*. Os campos IP *timestamp* e TCP *echo* foram usados para codificar o *nonce* e enviá-lo ao *firewall*. O *buffer* do *timestamp* deve ser marcado como *full* de modo, que os roteadores intermediários não manipulem o seu valor.

Em (LIEW et al., 2010) os autores relatam que até a realização de seu trabalho, somente duas implementações usam *One Time Password* (OTP). A primeira, o *Cerberus*[10] usa como OTP o *timestamp* (até precisão de minuto). A segunda, *COK*[11] aplica *hash* na senha do usuário iterativamente para criar uma cadeia de saídas. Os autores, também, notaram que não há garantia de que o cliente ao se conectar à porta aberta seja o mesmo cliente autenticado, pois não existe ligação entre eles. Propuseram um *framework* que envia OTP via SMS[12] ao usuário usando *Pseudo-Random Number Generator*[13] (PRNG) com o *timestamp* e uma porta randômica usada no acesso. A derivação de chaves, também, foi usada. A senha recebida é usada para adicionar confidencialidade e calcular MAC[14] do SPA. Também é criado o túnel *IPSec* VPN com os parâmetros passados via SPA. É relatado que pode haver limitações pontuais no projeto devido a possíveis modificações no protocolo *IPSec*.

No trabalho de (ZORKTA; ALMUTLAQ, 2012) o cliente envia um pacote TCP com as *flags* SYN/FIN/RST contendo os dados da validação criptografados. Usa o *timestamp* e um número randômico para prevenir *replay attacks*. Relata que seu trabalho resolve dois problemas: 1. ataque de reserva de recursos (através de atributo do SPA indicando porta do próximo SPA); 2. falta de associação entre o processo de autenticação e o processo de estabelecimento de seção (por meio de porta informada pelo SPA).

Em (KUMAR; TALWAR, 2012) é proposto um mecanismo nomeado de QUICKKNOCK, melhorando as potencialidades de tecnologias como *port knocking* e SPA usando *firewall* e criptografia. Com *steganografia*[15], usa os campos de número de sequência e *timestamp* do *header* TCP SYN para embutir o código *hash* resultante da autenticação. Se, a verificação for bem-sucedida, o servidor permitirá que a conexão continue, caso contrário, o pacote será descartado. Há uma limitação de 32 *bits* e, isso, limita o envio do *hash* completo, mantido às claras. O cliente e o servidor compartilham uma chave, bem como um contador (usado como *nonce*) que é incrementado para cada conexão do cliente. O SPA e a conexão TCP subsequente são tratados em um único pacote enviado.

---

[10] http://silverstr.ufies.org/blog/Cerberus.ppt

[11] http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-worth-up.pdf

[12] SMS é a sigla de *Short Message Service*, que em português significa Serviço de Mensagens Curtas. SMS é um serviço muito utilizado para o envio de mensagens de texto curtos, através de telefones celulares. É um serviço rápido e eficiente.

[13] São algoritmos para geração de números com propriedades semelhantes à dos números aleatórios (*random numbers*).

[14] Autenticador de mensagem (em inglês, *message authentication code*).

[15] É o estudo e uso das técnicas para ocultar a existência de uma mensagem dentro de outra, uma forma de segurança por obscurantismo.

Os estudos (TARIQ; BAIG; SAEED, 2008; LIEW et al., 2010; ZORKTA; ALMUTLAQ, 2012; KUMAR; TALWAR, 2012) propuseram soluções para criação do SPA e estabelecimento da seção TCP subsequente para autenticação do usuário.

A pontualidade foi abordada em (LIEW et al., 2010) e a originalidade em (TARIQ; BAIG; SAEED, 2008; LIEW et al., 2010; ZORKTA; ALMUTLAQ, 2012; KUMAR; TALWAR, 2012). A integridade de dados foi fortemente usada em (ZORKTA; ALMUTLAQ, 2012) e parcialmente (TARIQ; BAIG; SAEED, 2008; LIEW et al., 2010; KUMAR; TALWAR, 2012). A autenticidade foi abordada em (TARIQ; BAIG; SAEED, 2008; LIEW et al., 2010; ZORKTA; ALMUTLAQ, 2012; KUMAR; TALWAR, 2012). A derivação de chaves foi, somente, usada em (LIEW et al., 2010). Não houve considerações sobre a compatibilidade com o protocolo IPv6 e a confidencialidade de tráfego nos trabalhos.

O trabalho de (KUMAR; TALWAR, 2012) usa com *steganografia*, os campos do *header* TCP SYN para embutir o código *hash* resultante da autenticação, onde o SPA e a conexão TCP subsequente são tratados em um único pacote enviado. Já (LIEW et al., 2010) propôs um canal *out-of-band* (SMS) para receber dados para envio do SPA e criação da VPN subsequente. O estudo de (TARIQ; BAIG; SAEED, 2008) envia um *nonce* via SPA, em que o servidor somente aceitará o TCP SYN caso contenha o *nonce* embutido. Em (ZORKTA; ALMUTLAQ, 2012) o cliente envia dentro do SPA a porta em que irá se conectar via TCP.

Ao analisar os trabalhos (TARIQ; BAIG; SAEED, 2008; LIEW et al., 2010; ZORKTA; ALMUTLAQ, 2012; KUMAR; TALWAR, 2012), observou-se que os autores não abordaram todos os aspectos de segurança de forma equânime. Tal observação foi fundamental para adoção dos objetivos e metodologia nesse trabalho. Abordou-se o SPA com funcionalidades sólidas de pontualidade, originalidade, integridade e autenticidade de dados, derivação de chaves, compatibilidade com o protocolo IPv6 e confidencialidade de dados e de tráfego. Criou-se também uma nova arquitetura capaz de ocultar conexão a subsequente sem alterar as funcionalidades padrões do protocolo TCP/IP.

4.2 Projeto do Novo SPA

A *Internet* é um ambiente particularmente compartilhado, sendo usada por empresas concorrentes, governos mutuamente hostis e criminosos oportunistas. Para que os dados transmitidos não sejam comprometidos, segundo (PETERSON; DAVIE, 2013) é recomendável abordar os seguintes aspectos de segurança: confidencialidade de dados e de tráfego, integridade e autenticidade, originalidade e pontualidade.

4.2.1 Análise de Requisitos

Inicialmente, apresenta-se na Tabela 1 os recursos já propostos pelo protocolo SDP v1.0.

Tabela 1 – Recursos propostos pela v1.0 do SDP e objetivo de sua ação.

Recurso	Objetivo
Autenticidade	SPA
Originalidade	SPA

Fonte: o Autor.

Os recursos de autenticidade e originalidade propostos por (PETERSON; DAVIE, 2013) já fazem parte do protocolo SDP e tiveram suas ações fortalecidas e melhoradas.

Visando melhorias na segurança do *Single Packet Authorization,* projeta-se uma estrutura modular para sua criação. A Tabela 2 apresenta seis novos recursos propostos.

Tabela 2 – Novos recursos propostos e objetivos de sua ação.

Recurso	Objetivo
Pontualidade	SPA
Integridade	SPA
Confidencialidade de dados	SPA
Confidencialidade de tráfego	SPA
Compatibilidade IPv6	SPA e arquitetura de envio e recebimento
Derivação de chaves	*Password* do usuário

Fonte: o Autor.

Os recursos de pontualidade, integridade, confidencialidade de dados e de tráfego proposto por Peterson e Davie (2013) e a derivação de chaves proposta por Liew et al. (2010) são incorporados ao modelo. A compatibilidade IPv6 também foi incorporada ao padrão.

A Especificação v1.0 recomenda que todos os componentes devem ser projetados para fim de escalabilidade ou redundância. Neste sentido, os seguintes recursos funcionais são implementados:

- A arquitetura deve ser projetada para fornecer escalabilidade e redundância;

- O acesso ao *Controller* deve ser em IPv4 e IPv6;
- A infraestrutura deve ocultar a conexão TCP subsequente;
- O dispositivo deverá fazer parte do processo de autenticação;
- Canais *out-of-band* são criados.

4.2.2 Modelo Entidade Relacionamento

Definido os novos recursos, foi elaborado um Modelo Entidade Relacionamento (MER)[16] (vide Figura 5) para dar suporte às novas funcionalidades. Aborda-se também no modelo conceitual, a nova arquitetura de envio e recebimento do SPA, além da conexão TCP subsequente.

O modelo relacional foi desenvolvido para facilitar o projeto de banco de dados relacionais[17] permitindo a representação de todas as estruturas e um melhor diálogo entre os profissionais envolvidos no projeto. Suas maiores vantagens são: flexibilidade, adaptabilidade, simplicidade e objetividade.

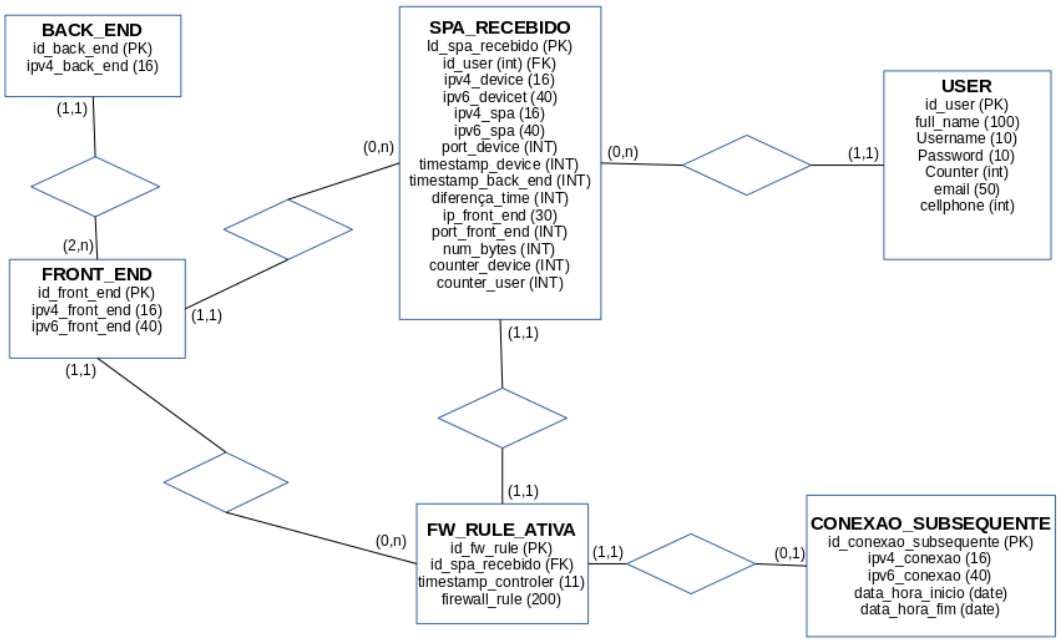
O *Controller back end* possui dois ou mais *Controllers front end* sob seu comando, desse modo, é possível escalonar *Controllers front end* sob demanda.

A cada *Controller front end é* permitido receber muitos SPA e possuir muitas regras ativas em seu *firewall*, ou seja, IPs que após a validação do SPA têm seu ingresso permitido no perímetro por um período configurável.

[16] É um modelo abstrato cuja finalidade é descrever, de maneira conceitual, os dados a serem utilizados em um sistema de informações. Descreve os objetos (entidades) envolvidos, com suas características (atributos) e como elas se relacionam entre si (relacionamentos).

[17] Se baseia no princípio de que todos os dados estão armazenados em tabelas (ou, matematicamente falando, relações).

Figura 5 – Modelo ER proposto para a nova arquitetura SDP.



Fonte: o Autor.

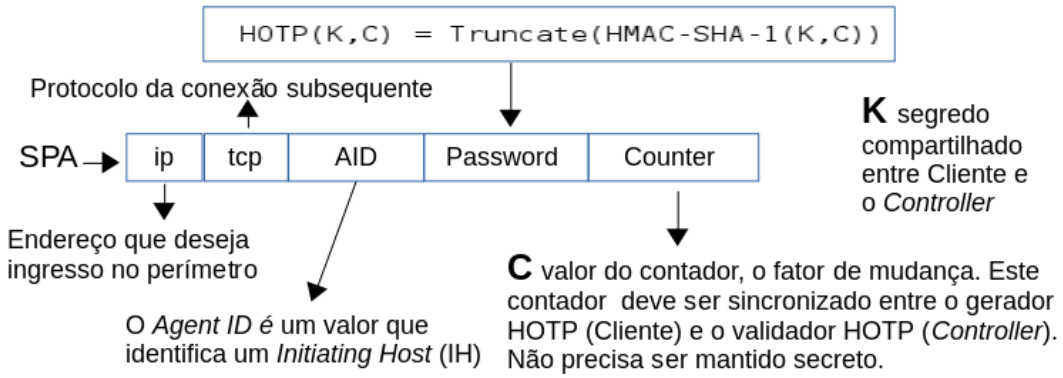
Para cada SPA recebido sempre se deve ter um usuário associado, um *Controller front end* e uma regra ativa permitindo o ingresso no perímetro.

A entidade “conexao_subsequente” apresenta um relatório de todos os acessos e o tempo de conexão em que o usuário esteve conectado.

4.3 Criação Modular do SPA

O envio do SPA é o primeiro passo para acesso a um SDP. A Figura 6 ilustra como o SPA é elaborado pela Especificação 1.0 (BILGER et al., 2014). Observa-se que o resultado do algorítimo HOTP gera um valor relativo ao campo *Password*; AID é um valor que identifica um Cliente; O protocolo TCP é definido como protocolo da conexão subsequente; IP é o endereço que deseja ingresso no perímetro; Já o *Counter* (C) introduz o conceito de *One Time Passwords* baseado em eventos e, (K) é a senha simétrica compartilhada entre o Cliente e o *Controller.*

Figura 6 – Formato do pacote SPA definido pela Especificação 1.0.



Fonte: Adaptado de Especificação 1.0 (BILGER et al., 2014).

Visando aumento dos níveis de segurança, são propostas a seguir melhorias na criação do SPA. A construção de soluções de segurança em módulos podem aumentar, consideravelmente, o grau de dificuldade de detecção, replicação ou leituras dos dados em um ambiente hostil como a *Internet*. Nesse sentido, incorporou-se ao modelo seis aspectos de segurança propostos por (PETERSON; DAVIE, 2013) e se acrescentou um padrão de derivação de chaves proposto por Liew et al. (2010). Também é implementado a compatibilidade com o protocolo IPv6. A seguir, os oito módulos são detalhados.

4.3.1 Módulo de Derivação de Chaves

Se as senhas usadas para criptografar e descriptografar pacotes SPA forem fracas, elas se tornam vulneráveis a ataques de dicionário e força bruta. Então, um invasor em um ataque *man-in-the-middle* pode facilmente capturar o pacote de autorização e obter a palavra secreta. Com a senha, ele pode descriptografar e usar as informações para criar seu próprio pacote (QASIM, 2012) .

Para evitar a exposição da chave mestra durante a transmissão do SPA é proposto o uso de funções de derivação de chaves (seção 2.4). Essa solução torna mais difícil a quebra de senhas. Com o uso da função PBKDF2, a senha mestra se torna uma chave derivada e alongada. Duas chaves derivadas são geradas, nomeadas de *Key1* e *Key2*.

De acordo com (KALISKI, 2000), o número mínimo recomendado de iterações para uso na função PBKDF2 é 1000. Como mencionado na seção (2.4), por ser um valor configurável, cada administrador de sistemas deve definir o tempo de processamento que irá impactar na autenticação do usuário, tendo em vista aumento dos custos contra ataques de força bruta ou dicionário. O valor das iterações da *Key1* e *Key2* é configurável e serve apenas para testes e demostração.

Tendo em vista que o número originalmente recomendado de iterações pela

função PBKDF2 é 1000, utilizou-se 1000 iterações para a *Key1.* Para distinção do *hash* resultante entre as chaves, utilizou-se 2000 iterações para *Key2*. A derivação de chaves *Key1* teve seu uso aplicado no módulo originalidade, autenticidade e integridade de dados conforme a função:

$$Key1^{(1000)} = (HmacSha256(pass \parallel salt1 \parallel 32))$$

Como exemplo, tem-se a senha "1234" alongada e derivada mil vezes com um *hash* resultante de 32 *bytes*:
Key1=7144818a81fa1f3909f5a9fc5239ab6b4c7d5913285be5b53b1fa573a6a2c5f6

No módulo de confidencialidade de dados, usou-se a derivação de chaves *Key2*, de acordo com a função:

$$Key2^{(2000)} = (HmacSha256(pass \parallel salt2 \parallel 32))$$

Com isso, a senha "1234" é alongada e derivada duas mil vezes e seu resultado é um *hash* de 32 *bytes*:
Key2=d771d476b97970c71e86ded3006a422a23623c5623b7b44e449b239833e3106e

4.3.2 Módulo Pontualidade

Um protocolo que detecta técnicas de atraso fornece pontualidade. É necessário incluir um leve carimbo de tempo[18] no SPA para prevenir o recebimento e processamento de pacotes desatualizados. Um atacante pode capturar um pacote SPA e transmiti-lo conforme a sua conveniência em um ataque posterior.

A Especificação 1.0 (BILGER et al., 2014) não abordou um atributo que forneça pontualidade. O campo *timestamp*, conhecido como *Current Unix Timestamp* - RFC 3339 (NEWMAN; KLYNE, 2002) foi incluído ao modelo para ser o fator de referência de tempo humano. O *Controller* recupera o valor do *timestamp* criptografado entre os outros campos do *payload* recebidos via SPA para confrontar com o seu intervalo permitido. Os objetivos do módulo pontualidade são:

- Detectar precisão de avanço ou atraso de tempo do relógio do Cliente em relação ao *Controller* com precisão de segundos para validar a originalidade do SPA;
- Descartar o SPA e não processar os módulos subsequentes caso o *timestamp* não esteja no intervalo de tempo permitido. A RFC 6238 (VIEW et al., 2011) orienta a divisão do tempo em 30 segundos. Nesse estudo, o *Controller* aceita o SPA se ele estiver dentro do intervalo permitido, até uma medida de tempo anterior e uma posterior do *timestamp*/30;
- Possibilitar ao usuário através do *canal out-of-band* visualizar possível erro de sincronia do relógio do seu dispositivo em relação ao *Controller*.

[18] É uma sequência de caracteres, indicando a data e ou o tempo em que um determinado evento ocorreu.

Caso o *timestamp* do Cliente esteja dentro do seu intervalo permitido, inicia-se o processamento do módulo originalidade.

4.3.3 Módulo Originalidade

No protocolo TOTP (VIEW et al., 2011) a combinação (*password* + *timestamp* = *hash* resultante) pode se tornar alvo de ataques. Um atacante pode originar ataques de força bruta ou dicionário sincronizados no tempo. Isso se deve ao fato de que o *timestamp* é de conhecimento público e a rotação do relógio do *Controller* não impede novos ataques. Já no protocolo HOTP (VIEW et al., 2005), uma senha criada com (*counter+password*) pode ter validade indeterminada.

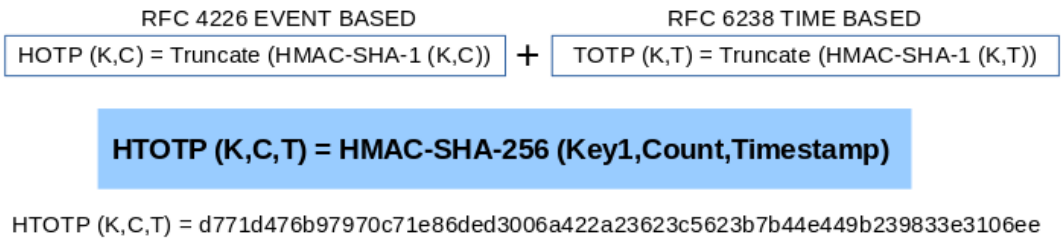
Com o objetivo de aumentar da segurança contra ataques de repetição, força bruta ou dicionário, esse trabalho uniu os protocolos HOTP e TOPT para criar a função **HTOTP**. Com isso, têm-se senhas de uso único baseado em eventos (*counter*) e tempo humano (*timestamp*). Incorporar um *counter* para cada usuário, sua senha simétrica e o valor do *timestamp*, torna ataques de força bruta, dicionário e principalmente *replay* praticamente ineficazes. Considerações importantes sobre a implementação do módulo originalidade:

- A RFC 4226 (VIEW et al., 2005) recomenda um valor máximo configurável da **janela de sincronização** que pode ser aceito quando há diferença entre os valores dos contadores de eventos do Cliente e do *Controller*. Cada implementação pode definir o valor da janela de sincronização, onde terá impacto no tempo de processamento do SPA*;*
- A RFC 6238 (VIEW et al., 2011) orienta a divisão do *timestamp* por 30 e recomenda que seja aceito um valor anterior e outro posterior ao tempo atual na validação do *timestamp* recebido;
- O *Controller,* ao receber os valores *counter* e *timestamp* do cliente verifica se ambos estão dentro dos seus intervalos permitidos. Caso não estejam o SPA é descartado, caso contrário o HTOPT resultante é gerado e comparado com o recebido.

No Cliente, a função HTOTP usa a chave derivada *Key1*, o valor do *counter* e *timestamp* para gerar o *hash* resultante.

No *Controller*, a função HTOTP usa o valor do *timestamp* e *counter* recebidos inclusos no SPA e a chave derivada *Key1* simétrica do Cliente da base de dados para novamente gerar o *hash* resultante para comparação. A Figura 7 apresenta os detalhes da nova função criada, nomeada HTOTP.

Figura 7 – Função HTOTP criada para fortalecer o módulo originalidade.



Fonte: o Autor.

A função HTOTP criada para fortalecer a originalidade gera um *hash* resultante de 32 *bytes* fortalecido pela função SHA-256 (implementadada através da biblioteca *openssl/sha.h*)[19]. A sua criação originou-se após a junção de atributos das funções HTOP e TOTP. O truncamento do *hash* em 4 *bytes*, apesar de ser recomendado pelas RFCs (4226 e 6238) para facilitar a manipulação pelo usuário, não foi usado visando maior segurança contra ataques de força bruta ou dicionário.

Deve-se ressaltar que o módulo originalidade considera o seguinte *nonce* para cada SPA e relativo a um único usuário:

user || counter || timestamp || Key1

4.3.4 Módulos Integridade e Autenticidade de Dados

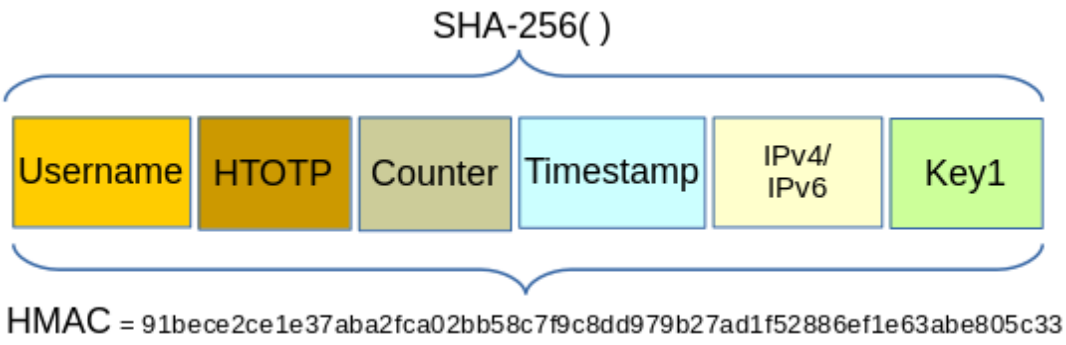
Um SPA vindo de um Cliente pode ter seu conteúdo modificado depois que ele o criou. Integridade e autenticidade são, de certa forma, inseparáveis (PETERSON; DAVIE, 2013). Um autenticador é um valor a ser incluído em um SPA transmitido, que pode ser usado ao mesmo tempo, para verificar autenticidade e integridade dos dados. Um autenticador pode combinar confidencialidade de dados e uma função de *hash* criptográfico HMAC. Ao contrário da função *hash* que fornece apenas verificação de integridade, um MAC também fornece verificação de autenticidade, assumindo que a chave secreta é conhecida apenas por entidades confiáveis (LIEW et al., 2010).

A Especificação 1.0 (BILGER et al., 2014) usa o *Secure Hash Algorithm* 1 (SHA-1) aplicado somente ao campo *counter* e *password* para prover Autenticidade de Dados do SPA. Recentemente, pesquisadores descobriram técnicas que encontram colisões de forma muito mais eficiente que a força bruta (STEVENS et al., 2017). Com o objetivo de melhorar a segurança é proposto o uso do *Secure Hash Algorithm 2 (*SHA-256) para criptografia do *payload*.

Nesse trabalho, a Integridade e Autenticidade de Dados é aplicado aos seguintes campos do *payload*: *username*, HTOPT, *counter, timestamp* e IPv4/IPv6. A chave derivada *Key1* é o valor secreto unido aos campos para formar o *hash* resultante. A Figura 8 apresenta o modelo proposto.

[19] Disponível em: https://github.com/openssl/openssl/blob/master/include/openssl/sha.h

Figura 8 – Aplicação de integridade e autenticidade aos atributos do SPA.



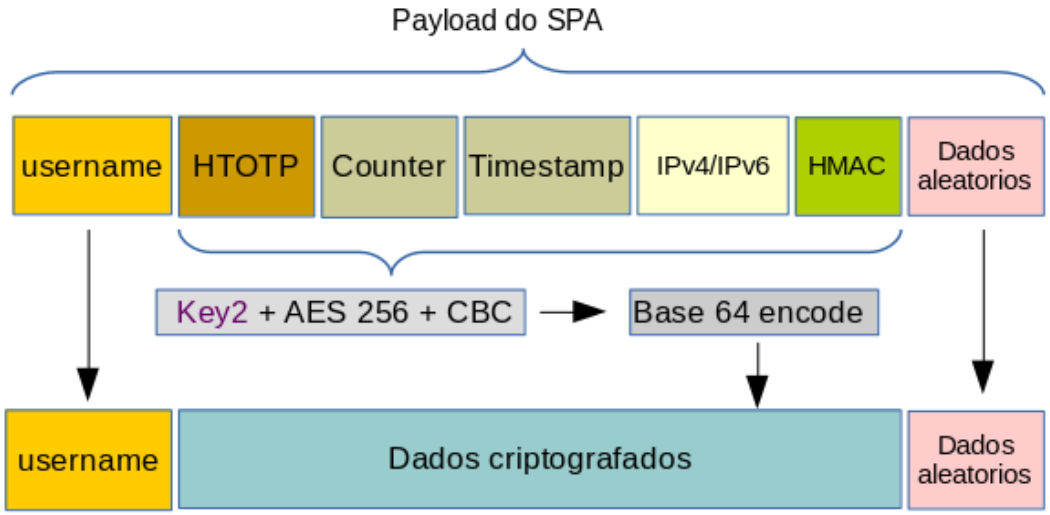
Fonte: o Autor.

O *hash* resultante da função SHA-256 (biblioteca *openssl/sha.h*) retorna um HMAC de 32 *bytes* que será novamente incorporado ao *payload* para ser criptografado com os outros campos.

4.3.5 Módulo de Confidencialidade de Dados

Na atual abordagem da Especificação 1.0 (BILGER et al., 2014), o SPA é enviado às claras, a qual os campos IP, TCP, AID, *Password* e *Counter* podem ser facilmente descobertos. No entanto, é prático e possível cifrar os dados transmitidos de modo a impedir que um atacante entenda o conteúdo. Algorítimos criptográficos baseados em cifras de chave simétricas foram utilizados, conforme a Figura 9.

Figura 9 – Aplicação da criptografia ao *payload* do SPA.



Fonte: o Autor.

O formato do SPA definido por esse trabalho apresenta sete campos. O campo *username* é enviado às claras, usado para identificar o Cliente quando o SPA for

recebido pelo *Controller*. O campo *dados_aleatorios* é usado pelo módulo de confidencialidade de tráfego. Os campos criptografados foram: HTOTP (*hash* resultante OTP), *counter* (contador baseado em eventos), *timestamp* (tempo humano), IPv4/IPv6 (endereço IP que deseja ingressar no perímetro) e HMAC (*hash* resultante da integridade e autenticidade de dados).

Na abordagem com cifra de chave simétrica, tanto o Cliente como o *Controller* compartilham a mesma chave para cifrar e decifrar. Ao longo dos anos, o Padrão Avançado de Cifração (AES - *Advanced Encription Standard*) tornou-se um padrão de cifra para chave simétrica. Chamado originalmente de *Rijndael* (DAEMEN; RIJMEN, 1999), admite tamanho de chaves de 128, 192 e 256 bits e encadeamento de bloco de cifra (CBC - *Cipher Block Chaining*) com tamanho de 128 bits (PETERSON; DAVIE, 2013; DESAI et al., 2013). O AES permite implementações rápidas em *software* e *hardware* e não exige muita memória.

O resultado da criptografia (caracteres não imprimíveis), geralmente, apresenta falhas quando transmitidos em protocolos projetados para lidar com dados textuais como o TCP ou UDP (JOSEFSSON, 2006). Nesse sentido, foram convertidos em *Base64*[20] pela aplicação Cliente para encaminhamento via *datagrama* UDP.

4.3.6 Módulo Compatibilidade IPv6

A compatibilidade com o protocolo IPv6 foi implementada para que toda e qualquer funcionalidade executada em IPv4 deve ser estendida ao protocolo IPv6. Nesse cenário, o *Controller* implementa o conceito de pilha dupla ou *dual stack*[21], assim exemplificado:

udp 0 0 200.18.45.119:8888 0.0.0.0:*
udp6 0 0 2804:0:4000:4018::119:8888 :::*

O protocolo IPv6 apresenta como principal característica o aumento no espaço para endereçamento. No IPv4 o campo do cabeçalho reservado para o endereçamento possui 32 *bits*, já no IPv6 o espaço para endereçamento é de 128 *bits*. O endereço IP a ser autorizado a ingressar no perímetro é transportado dentro do *payload*. Caso o endereço for IPv6 pode haver um aumento de até 96 *bits* no tamanho do *payload.*

Diversas mudanças estruturais foram feitas no desenvolvimento das aplicações Cliente e *Controller* para acomodar o novo endereçamento IPv6. As aplicações de *firewall* (do *Controller*) também tiveram que ser tratadas de forma distinta devido à diferença de sintaxe e operação das regras IPv4/IPv6.

[20] Esquema de codificação comumente usados quando há necessidade de codificar dados binários que precisam ser armazenados e transferidos em mídias projetadas para lidar com dados textuais. Disponível: https://www.base64decode.org/

[21] Consiste na convivência do IPv4 e do IPv6 nos mesmos equipamentos, de forma nativa, simultâneamente. Disponível: http://ipv6.br/post/transicao/

4.3.7 Módulo Confidencialidade de Tráfego

Avançando no conceito de confidencialidade, o ato de ocultar a quantidade de dados, origem ou destino é chamado de confidencialidade de tráfego. Saber a quantidade de dados que está indo para um determinado destino pode ser útil para um adversário em algumas situações. A Especificação 1.0 (BILGER et al., 2014) prevê o envio de um único pacote implementado pelo protocolo TCP. As seguintes funcionalidades que visam a confidencialidade de tráfego foram adicionadas ao modelo:

- Adicionou-se ao *payload* o campo *dados_aleatorios*, formado por caracteres de *Base64* de tamanhos variados gerados aleatoriamente (entre 0 e 100 *bytes*). O SPA passa a ter tamanhos variados dificultando estabelecimento de um padrão. Esses dados são descartados pelo *Controller*;
- Geralmente em um mecanismo SPA, são usados protocolos UDP, TCP ou ICMP (ZORKTA; ALMUTLAQ, 2012). Este trabalho usa o protocolo UDP (*User Datagram Protocol*). É um protocolo não orientado a conexão da camada de transporte do modelo TCP/IP. Estabelece-se aqui o princípio de pacote único;
- O *Controller* deve ser identificado somente pelo seu endereço IP. O Cliente não efetua nenhuma resolução DNS para enviar o SPA;
- Não permitir que o *Controller* responda a qualquer requisição, dando a ideia de buraco negro. Somente SPA enviados a uma porta específica serão recebidos e processados pelo *Controller* e nenhuma resposta deve ser retornada;
- Alternar o envio do SPA e TCP SYN da conexão subsequente através dos protocolos IPv4 e IPv6 desde que o Cliente implemente pilha dupla.

Abaixo um trecho de *log* coletado via *Tcpdump*[22] das funcionalidades acima incorporadas ao padrão de envio de um SPA relativos ao módulo de confidencialidade de tráfego, exemplificados em quatro SPA enviados:

IP 191.32.114.196.33510 > 200.18.45.122.8888: UDP, length 315
IP 191.32.114.196.46719 > 200.18.45.122.8888: UDP, length 373
IP 191.32.114.196.44792 > 200.18.45.122.8888: UDP, length 289
IP 191.32.114.196.55532 > 200.18.45.122.8888: UDP, length 358

A porta UDP selecionada para envio do SPA é a 8888 (valor customizável). Após o recebimento e validação do SPA, o *Controller* permite que o dispositivo Cliente conecte-se via TLS mútuo a porta 443 dando início ao processo de autenticação do usuário.

[22] Ferramenta utilizada para monitorar os pacotes trafegados numa rede de computadores. Disponível em: http://www.tcpdump.org/

4.4 Conexão TCP Subsequente

A conexão TCP subsequente a ser estabelecida após o envio do SPA é um problema que ainda persiste em ambas as técnicas de autenticação (SPA ou *port knocking*), conforme mencionado na seção de trabalhos relacionados. Esta seção aborda o estabelecimento de conexão TCP subsequente entre o *Initiating Host* (Cliente) e o SDP *Controller.*

Nesse trabalho, não foram transportados dados no *header* dos protocolo TCP conforme relatado nos trabalhos relacionados de (TARIQ; BAIG; SAEED, 2008; KUMAR; TALWAR, 2012) pois não se deseja alterar suas funcionalidades padrões. Em relação ao *Controller*, não houve pré-alocação de recursos (e.g., portas TCP do *Controller front end* pré-alocadas para Clientes específicos conforme os trabalhos relacionados de (LIEW et al., 2010; ZORKTA; ALMUTLAQ, 2012)). Contudo, desenvolveu-se uma arquitetura escalável, redundante e com dupla abordagem (IPv4 e IPv6).

Para evitar que um invasor se conecte a um *Controller* em nome de um Cliente válido após a validação do SPA, a arquitetura SDP foi dividida em seguintes sistemas: dois *Controllers front end* com as mesmas funcionalidades, um *Controller back end* e um modelo de dados centralizado com consistência sequencial[23]. Com isso, a infraestrutura fornece escalabilidade e redundância ao modelo proposto além de ocultar a conexão TCP subsequente.

Como referenciado na seção (2.2), o SPA é usado para inicializar a comunicação nos seguintes casos: IH-Controller, AH-Controller, e IH-AH. Neste trabalho é proposto alterações nas formas de comunicações:

- O *Controller front end* atuará como um *gateway* entre os *hosts* de iniciação - IH (Clientes) e *host* de aceitação - AH (servidores, aplicações, redes e serviços);
- Não haverá mais comunicação direta entre os *hosts* de iniciação e os *hosts* de aceitação, tudo será a partir de um *gateway* (*Controller front end*);
- Os *hosts* de aceitação estarão sempre ligados aos *gateways* disponíveis e seus serviços serão oferecidos como um Catálogo de Serviços[24] aos usuários do SDP (*hosts* de iniciação - IH);
- Replicação dos *Controllers front* end em dois ou mais;
- O *Controller back end* é responsável pela administração e gerência dos *Controllers front end.*

A replicação dos *Controllers* propicia:

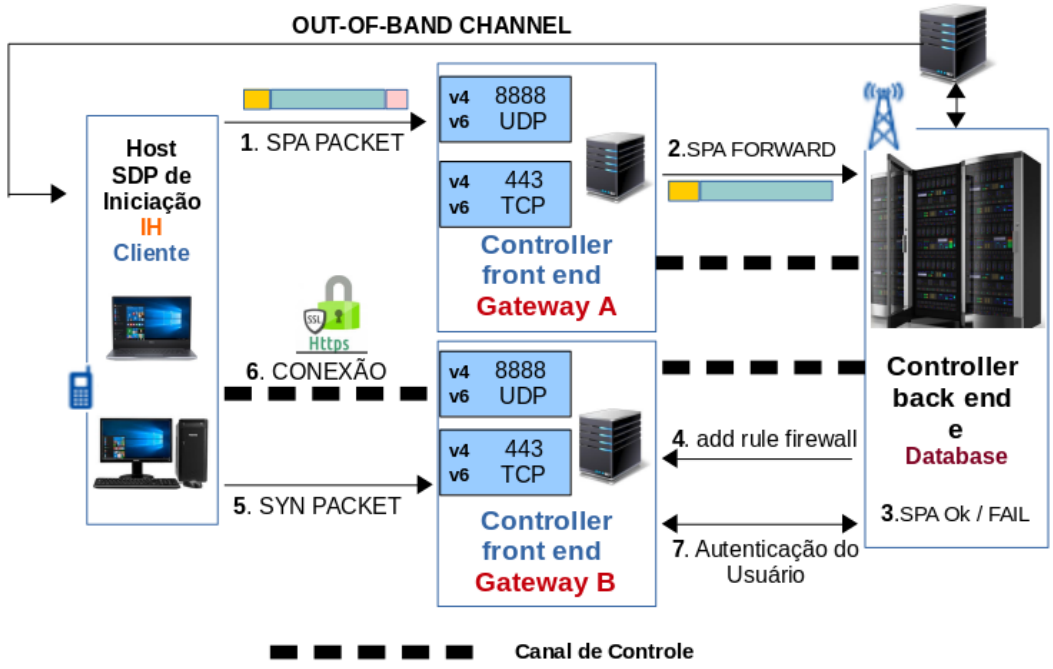
[23] Todas as operações são realizadas de acordo com uma ordem e nenhuma operação é iniciada antes do término da operação anterior.

[24] O objetivo do Catálogo de Serviços é trazer uma visão clara de quais serviços são ofertados.

- **Escabilidade:** refere-se a possibilidade de aumentar o processamento de operações simultâneas (recebimento de SPAs e conexões TCP subsequentes) mediante a adição de novos *Controllers front end*.
- **Redundância:** envolve técnicas de tolerância a falhas, tanto de *hardware* como de *software*. Com isso, os *Controllers front end e back end* podem ser replicados em seus componentes;

O SPA é o primeiro passo para ingresso em um SDP, nesse sentido é necessário que um aplicativo seja instalado no dispositivo Cliente. Este trabalho propõe o desenvolvimento de um aplicativo para efetuar o envio do SPA e informar ao usuário em qual *Controller front* end deve se conectar via *https* para efetuar sua autenticação. A Figura 10 apresenta o projeto desenvolvido.

Figura 10 – Nova arquitetura SDP proposta para envio do SPA, estabelecimento da conexão TCP subsequente e autenticação do Usuário.



Fonte: o Autor.

O detalhamento e funcionamento da arquitetura possui sete principais etapas descritas a seguir:

1) SPA *PACKET*: inicialmente o Cliente efetua o *download* do aplicativo cliente via conexão autenticada (canal *out-of-band*) através de endereço *web* previamente informado a ele. O aplicativo cliente possui uma lista ordenada de IPs dos *Controllers front end* que se pode conectar. O envio do SPA tem como porta destino 8888 (UDP) e é intercalado entre os *Controllers front end;*

2) SPA (*FORWARD*): após o recebimento, verifica-se se o *payload* contém os 3 atributos previstos (*username*, *dados_criptorados* e *dados_aleatorios*). Caso sim, o atributo *dados_aleatorios* é retirado do *payload* e o restante dos atributos são encaminhados ao *Controller back end* para validação;

3) SPA OK/*FAILURE*: o *Controller back end* recebe o SPA para processamento. Se as condições de todos os módulos de segurança forem satisfeitas o SPA é validado, caso contrário é descartado;

4) ADD *RULE FIREWALL*: o *Controller back end* solicita ao *Controller front end* subsequente (a conexão TCP subsequente é estabelecida em *Controller front end* alternado ao envio do SPA) para permitir o acesso à porta 443 TCP do IP informado pelo SPA;

5) SYN *PACKET*: alguns segundos após o envio do SPA o dispositivo Cliente já pode estabelecer conexão *https* com o segundo *Controller front end* de sua lista. Para isso, o *Controller front end* aguarda por um período configurável pelo *SYN PACKET* do Cliente e inicia o *handshake*[25] TLS. Não é permitido um *Controller front end* receber um SPA e estabelecer a conexão *https* subsequente para autenticação de um mesmo Cliente;

6) CONEXÃO *https*: após o *handshake* TLS, o Cliente recebe uma página para informar suas credenciais (*username* e *password*) e tem novamente um tempo pré-definido para finalizar a autenticação;

7) *USER AUTH*: o *Controller back end* recebe as credenciais do Cliente e dá continuidade ao processo de autenticação conforme a sua política de acesso.

Complementarmente, canais *out-of-band* são disponibilizados ao usuário para prover serviços como: lista de *Controllers front end* atualizada, *download* do aplicativo para envio do SPA, verificação do sucesso ou falha no envio e recebimento do SPA, políticas de uso, resolução de problemas, recuperação e alteração de senhas. Todos os SPA recebidos pelo *Controller back end* validados ou não, tem suas informações guardadas em uma base de dados. Todos os usuários têm seus dados (nome, telefones, endereço, tipo de vínculo com a empresa/organização, níveis e permissões de acesso, etc.) já anteriormente cadastrados na base de dados.

[25] O *handshake* SSL ou TLS permite que o cliente (portador do certificado público) e o servidor (de posse do certificado privado) SSL ou TLS estabeleçam as chaves secretas com as quais eles se comunicam.

4.5 Experimentos e Resultados

Esta seção apresenta os resultados obtidos através da avaliação da abordagem proposta. Inicialmente, experimentos foram realizados em dois conjuntos de testes relativos a construção modular do SPA: quantitativo e latência.

Nos experimentos quantitativos foram medidos os tempos de processamento em milissegundos à execução de aplicações de testes:

1) Dos módulos acoplados;

2) Avaliação do desempenho da função HTOTP criada.

O teste de latência apresenta a variação do *delay* tendo em vista a quantidade de *bytes* transportados entre o Cliente e o *Controller front end* conforme o acoplamento dos módulos.

As aplicações de testes foram desenvolvidas na linguagem de programação ANSI C e se dividem em 3: Cliente, *Controller Front end* e *Controller back end.* O *Controller back end* usa banco de dados *Mysql Server*[26] versão 5.5.9999 e o *Controller front end* utiliza o filtro de pacotes *Netfilter*[27] *(Iptables).*

De modo a auxiliar na obtenção dos valores quantitativos se desenvolveu uma função em ANSI C que realiza pontos de marcação do tempo no início e fim do processamento de cada módulo. Na análise da latência se utilizou a ferramenta *ping*[28]. Em cada módulo foram realizadas 30 repetições para cada conjunto de testes.

Os experimentos nos *Controllers back end* foram realizados em ambiente de virtualização *Xen Source*[29] com máquinas virtuais paravirtualizadas *Debian* 9 de 64 *bits,* 1024 MB de memória RAM e quatro processadores *Intel*(R) *Xeon*(R) 1.86GHz. Já o dispositivo Cliente é um *notebook* com 4 GB de memória RAM e quatro processadores Intel(R) Core(TM) 2.60GHz.

Para realizar o envio dos dados para o *Controller front end*, a aplicação Cliente utiliza um *socket* UDP. Para facilitar esse envio, as informações são encapsuladas em único pacote, utilizando-se a biblioteca *Binn*[30], a qual permite serializar diversas informações em um único pacote a ser enviado.

4.5.1 Análise Quantitativa dos Módulos

O parâmetro mais crítico para estudo analítico do envio e processamento do SPA, segundo (ZORKTA; ALMUTLAQ, 2012; TARIQ; BAIG; SAEED, 2008) é o fator de desempenho.

[26] Disponível em: https://www.mysql.com/
[27] Disponível em: https://www.netfilter.org/
[28] Disponível em: https://linux.die.net/man/8/ping
[29] Disponível em: https://www.xenproject.org/
[30] Disponível em: https://github.com/liteserver/binn

A sobrecarga de processamento foi avaliada se baseando na quantidade de tempo em milissegundos, à medida que cada módulo é processado no Cliente e no *Controller back end*. O custo de processamento do *Controller front end* é desprezível e não se constatou a necessidade de medi-lo. A Tabela 3 apresenta os resultados.

Tabela 3 – Resultados da simulação quantitativa em milissegundos relativos ao acoplamento dos módulos do SPA relativo a um único SPA.

Módulos Acoplados	**Cliente**		**Controller**	
	Indiv.	Total	Indiv.	Total
SPA básico[a]	0,26	0,26	0,49	0,49
Pontualidade	0,12	0,38	1,11	1,60
Originalidade	0,06	0,44	1,43	3,03
Integrid. e Autenticid.	0,10	0,54	0,78	3,81
Confidenc. de dados	2,42	2,96	2,78	6,59
Derivação *Key1*[b] (1000 iterações)	2,21	5,17	2,19	8,78
Derivação *Key2*[c] (2000 iterações)	4,08	9,25	4,58	13,36
Compat. IPv6	0,09	9,34	0,08	13,44
Confidenc. de tráfego	0,03	**9,37**	0,00	**13,44**

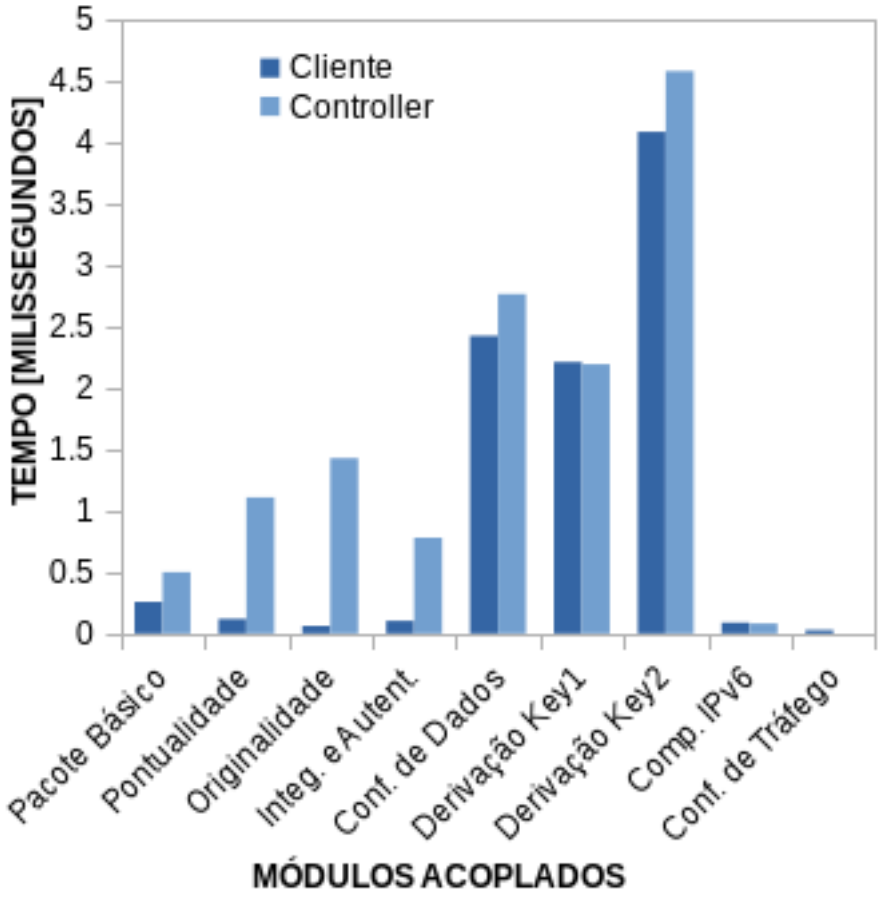
Fonte: o Autor.

[a] Conforme Figura 6 - Formato do pacote SPA definido pela Especificação 1.0.
[b] Aplicada em: originalidade, integridade e autenticidade de dados.
[c] Aplicada em: confidencialidade de dados.

A tabela apresenta os resultados dos tempos de processamento total e individual obtidos durante o acoplamento dos módulos no dispositivo de teste Cliente e no *Controller back end.* O tempo total é incrementado à proporção que cada módulo é processado. O tempo individual representa o custo de processamento de cada módulo.

A Figura 11 sintetiza os resultados da tabela anterior mostrando a carga de processamento individual de cada módulo à medida que ocorre a criação do SPA pelo Cliente e a validação do SPA pelo *Controller back end*. Os resultados da simulação são relativos a um único SPA.

Figura 11 – Tempo do processamento individual de cada módulo.

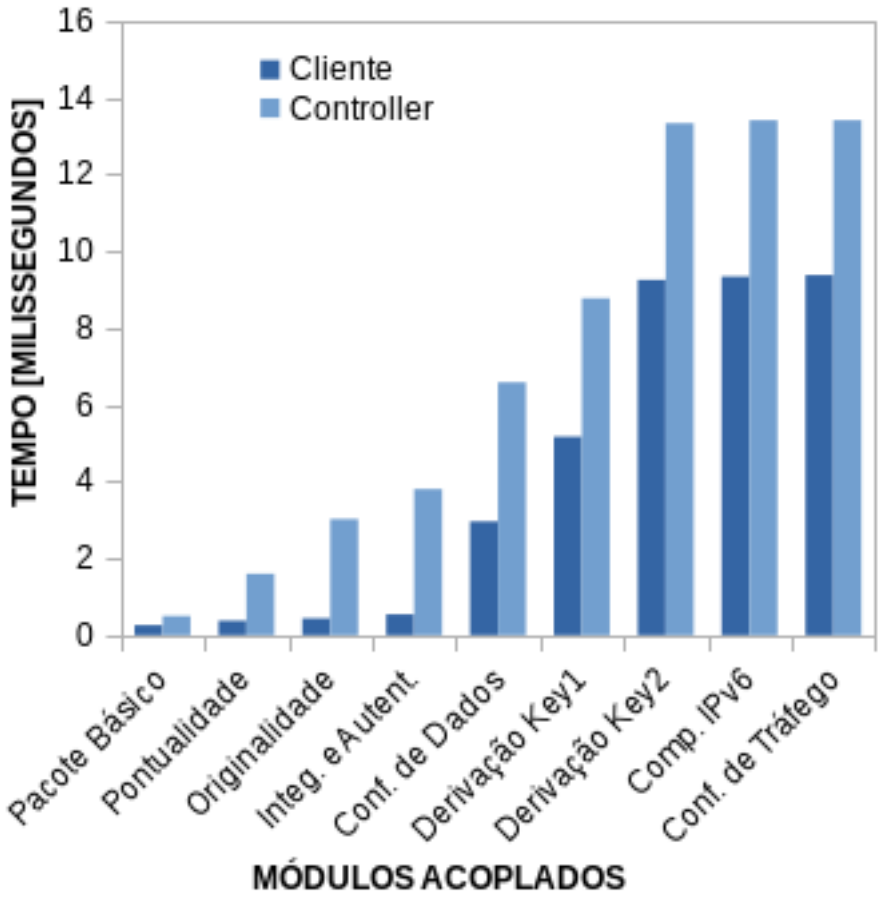


Fonte: o Autor.

Observa-se que a derivação de chaves *Key2* (2000 iterações) apresenta o maior tempo de processamento que um módulo alcançou pelo modelo proposto: 4.08 ms pelo Cliente e 4.58 ms pelo *Controller back end*. O número de iterações a ser usado está ligado diretamente ao custo de processamento e memória. O valor das iterações é configurável, tendo em vista o aumento dos custos contra ataques de força bruta ou dicionário.

Para demonstrar o tempo de processamento total, enquanto os módulos são processados, apresenta-se a Figura 12 com o somatório em milissegundos. A proporção que os módulos foram incluídos, observa-se o aumento do tempo de processamento. O crescimento do processamento se manteve constante até a derivação *Key2*. Compatibilidade IPv6 e confidencialidade de tráfego apresentaram processamentos não significativos.

Figura 12 – Somatório dos tempos de processamento dos módulos.



Fonte: o Autor.

Constata-se que o tempo de processamento dos módulos pelo *Controller back end* variam de 0.49 até 13.44 ms. Pelo Cliente oscilam entre 0.26 e 9.37 ms. O tempo gasto em processamento é compensado pelo aumento significativo dos níveis de segurança do SPA.

4.5.2 Análise da Função HTOTP

Na definição de projeto, optou-se por enviar os atributos *counter* e *timestemp* junto com os demais campos do *payload*, todos criptografados. O objetivo dessa forma de implementação é diminuir o tempo de processamento do SPA. Com os valores de *counter* e *timestamp* já validados dentro dos intervalos permitidos pelo *Controller back end, o hash* resultante é gerado uma única vez (1x).

Considerando que os atributos *counter* e *timestemp* não fossem transportados pelo SPA teríamos os seguintes cenários:

1) Caso o relógio do Cliente estivesse dessincronizado, o *Controller back end* aceitaria uma medida de tempo posterior e uma anterior, podendo gerar até três

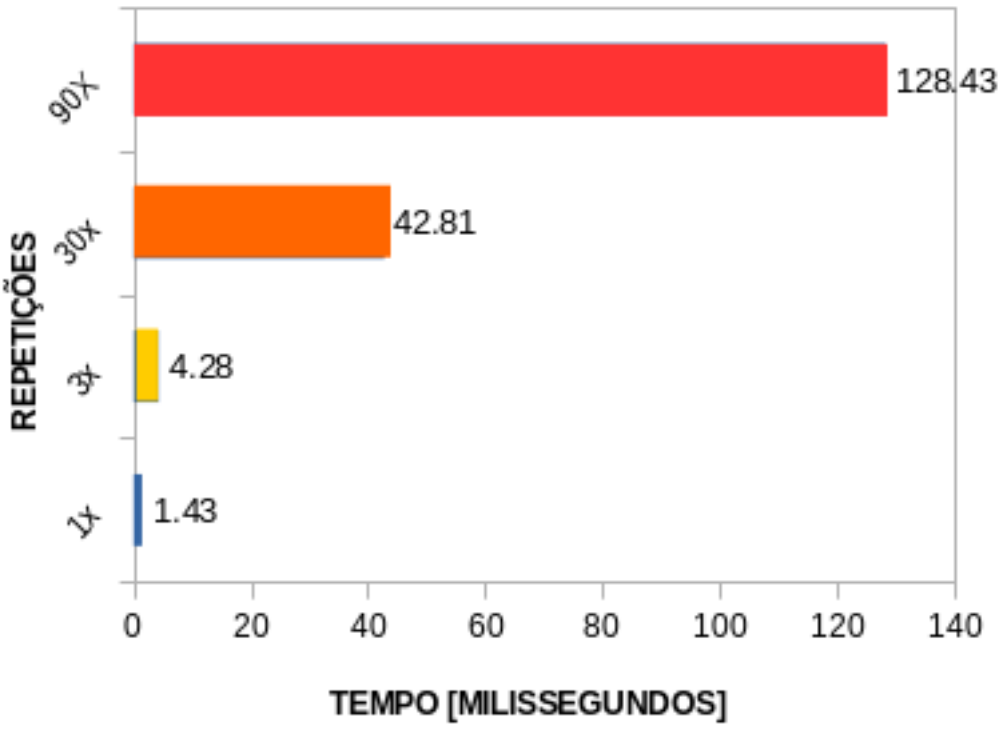
vezes (3x)[31] o *hash* resultante para validação ou descarte do pacote;

2) Já com o valor do *counter* dessincronizado, o *hash* resultante poderá ser gerado até trinta vezes (30x)[32] antes de aceitar ou rejeitar o SPA;

3) Supondo que os valores do *counter* e do *timestamp* estivessem desatualizados, o *Controller* terá que testar todas as possíveis combinações até noventa vezes (90x) antes de aceitar ou descartar o SPA.

A Figura 13 apresenta a variação do tempo de processamento de um único SPA. Ele poderia variar de 1.43 até 128.43 ms.

Figura 13 – Gráfico resultante da função HTOTP.



Fonte: o Autor.

Com base nos resultados, observa-se que o transporte dos campos *timestamp* e *counter* dentro do *payload* SPA oferece expressivo ganho no tempo de processamento, dado que se tem a certeza de que o SPA será processado uma única vez (1x).

[31] Valor configurável da janela de sincronização que pode ser aceito da variável *timestamp*. A RFC 6238 orienta a divisão do *timestamp por* 30 e recomenda que seja aceito um valor anterior e outro posterior ao tempo atual na validação da senha.

[32] Valor configurável da janela de sincronização que pode ser aceito quando há diferença entre o valor da variável *counter* de cada usuário e o valor salvo no *Controller.*

4.5.3 Análise da Latência

Analisou-se o *delay*[33] em milissegundos tendo em vista a quantidade de dados em *bytes* transportados do Cliente até o *Controller front end*. A Tabela 4 apresenta os resultados dos testes realizados em IPv4 em rede local *gigabit.* Os testes IPv6 não foram apresentados devido à similaridade de valores ao IPv4. Os resultados não utilizaram a sincronia de relógios entre o Cliente e o *Controller front end*, apenas é obtido o tempo de ida e volta de um pacote ICMP e dividido por 2, estimando-se assim o tempo de ida do pacote UDP.

Tabela 4 – Resultados do teste de latência.

Módulos Acoplados	***Payload* (*bytes*)**	***Delay* (ms)**
SPA Básico[a]	42	0,115
Pontualidade	88	0,132
Originalidade	97	0,135
Integridade e Autenticidade	185	0,141
Confidenc. de Dados	275	0,145
Derivação *Key1*[b]	275	0,145
Derivação *Key2*[c]	275	0,145
Compat. IPv6 (+12 *bytes*)	275∼287	0,145
Confidencialidade Tráfego	275∼375	0,146

Fonte: o Autor.

[a] Conforme Figura 6 - Formato do pacote SPA definido pela Especificação 1.0.
[b] Aplicada em: originalidade, integridade e autenticidade de dados.
[c] Aplicada em: confidencialidade de dados.

Os resultados apresentam a evolução em *bytes* do *payload* e o *delay* correspondente em milissegundos a cada módulo incorporado ao modelo. O crescimento do *payload* se mantém linear até o módulo de confidencialidade de dados e constante na derivação *Key1*, *Key2* e compatibilidade IPv6. Quando o *payload* transportar um endereço IPv6 para acesso na conexão TCP subsequente, poderá haver um acréscimo de até 12 *bytes*. Na confidencialidade de tráfego, o *payload* tem um acréscimo entre 0 e 100 *bytes* podendo variar entre 275 a 375 *bytes.*

O teste de latência mostra que o acréscimo do *delay,* tendo em vista o aumento dos *bytes* do *payload,* não apresenta atrasos significativos. Diante disso, é completa-

[33] Significa atraso e representa a diferença de tempo entre o envio do SPA e o recebimento pelo *Controller front end*.

mente viável o aumento do tamanho do *payload* em prol do aumento significativo dos níveis de segurança.

4.6 Discussões

Os resultados experimentais demonstraram aspectos relacionados a construção e envio um *Single Packet Authorization* projetado sob o aspecto da pontualidade, originalidade, integridade e autenticidade de dados, derivação de chaves, compatibilidade com o protocolo IPv6 e confidencialidade de dados e de tráfego. A construção de soluções de segurança em módulos aumentam consideravelmente o grau de dificuldade de detecção, replicação ou leituras dos dados de um SPA. Cada vez que ocorre a iminência de um ataque, há mecanismos que fazem com que os níveis de segurança sejam fortalecidos, diminuindo sua probabilidade de êxito.

Os resultados do experimento quantitativo mostram que a adição dos módulos propostos são viáveis. O aumento do tempo de processamento de um SPA se justifica pelos ganhos significativos de proteção, demonstrados no capítulo 6.

As conclusões da experimentação do teste de latência revelam que o *delay* imposto pelo acréscimo de *bytes* adicionado ao *payload* pelos módulos geram impactos poucos significativos.

Cada tentativa de ataque de força bruta ou dicionário pode ter a mesma probabilidade de sucesso, independentemente da rotação do relógio. A RFC 6238 (VIEW et al., 2011) orienta a divisão do tempo em 30 segundos e o *timestamp* é de conhecimento público. Assim, a combinação (*password* + *timestamp*) pode ser alvo de ataques. No protocolo HOTP, caso o *hash* resultante for comprometido, o seu tempo de validade é indeterminado. A junção dos protocolos TOTP e HOTP resultaram na função HTOTP. Esta função une o tempo humano e um contator individual para fortalecer a originalidade das senhas.

A análise da função HTOTP mostra que o transporte criptografado dos campos *timestamp* e *counter* pelo SPA garante que o *Controller back end* gere o *hash* resultante uma única vez para validação, podendo reduzir em até 90 vezes o tempo de processamento do módulo originalidade.

Com o uso de *Key Derivation Function* pode-se gerar uma ou mais chaves derivadas e alongadas a partir de uma chave mestra (senha secreta). A derivação de chaves *Key1* teve seu uso aplicado no módulo originalidade, autenticidade e integridade de dados. No módulo de confidencialidade de dados, usou-se a derivação de chaves *Key2*. Com isso, a chave mestra não foi exposta durante a transmissão do SPA. Além disso, as iterações fortalecem proteção contra ataques de força bruta ou dicionário.

Evitar que um invasor se conecte a um *Controller front end* em nome de um Cliente válido após a validação do SPA e antes do estabelecimento do TLS mútuo foi um grande desafio abordado por esse trabalho. Organizar a infraestrutura em dois *Controllers front end* e um *Controller back end*, além de um canal *out-of-band,* fornece escalabilidade e redundância ao modelo proposto. Alternar o envio do SPA e da conexão TCP subsequente entre os *Controllers front end* propicia a ocultação da conexão TCP subsequente. Também é possível intercalar o envio do SPA entre os protocolos IPv4 e IPv6, desde que o Cliente implemente o protocolo IPv6.

4.7 Conclusão Parcial

Essa seção analisou a Especificação 1.0 (BILGER et al., 2014) e propôs outras definições ao padrão proposto pela *Cloud Security Alliance,* tendo como objetivo a definição de um novo modelo de criação e envio de um *Single Packet Authorization*, passo inicial para ter acesso a um Perímetro Definido por Software. Complementarmente se definiu uma nova arquitetura para estabelecimento do TLS mútuo para autenticação do Cliente.

O SPA não é um substituto para autenticação. Ele é apenas outra camada que permite que IPs estejam acessíveis e portas TCP/UDP sejam abertas somente quando necessário.

Quando o SPA é recebido pelo *Controller front end* ocorre uma autenticação, onde validando os módulos do SPA, permite-se que o endereço IP do dispositivo Cliente possa ingressar no SDP. A segunda etapa da autenticação do usuário ocorre após estabelecer a conexão TPC subsequente, onde novamente é informado o *username* e *password* via conexão *https*.

Por meio da avaliação experimental quantitativa e teste de latência, verificou-se o aumento significativo nos níveis de proteção e confiabilidade do modelo proposto não existindo, no entanto, uma solução única. Constatou-se que é necessário várias tecnologias e processos que ofereçam ocultação da infraestrutura, de aplicações e controles de acesso, bem como prover conexão segura para proteger redes e servidores de ataques cibernéticos. As soluções propostas contribuem para o aumento dos níveis de proteção e de resiliência do padrão de referência SDP.

Alternar o envio do SPA e da conexão TCP subsequente entre os *Controllers front end* propicia a ocultação da conexão TCP subsequente. No entanto, observou-se que os problemas ainda persistem, tendo em vista falha no modelo TCP/IP quando a identidade de um dispositivo é vinculado ao seu endereço IP.

5 AUMENTANDO OS NÍVEIS DE SEGURANÇA NA AUTENTICAÇÃO COM *DEVICE FINGERPRINTING*

No modelo de comunicação TCP/IP é assumido que apenas um dispositivo acessará um servidor de um determinado endereço. A tradução de endereços de rede (do inglês *Network Address Translation* - NAT), *proxy* e técnicas similares podem ter um mesmo IP compartilhado centenas ou milhares de vezes, bem como ataques de IP *spoofing* podem ocorrer. Portanto, nesse cenário, não há certeza de que o computador que deseja acesso ao perímetro seja um computador autorizado, ou seja, o mesmo que teve sua autenticação validada e sua autorização permitida via SPA. Diante do exposto, concluíu-se que somente a ocultação da conexão TCP subsequente apresentada no capítulo 4 não resolve o problema de vincular uma identidade a um endereço IP.

Este capítulo apresenta uma extensão à arquitetura SDP para resolver falhas que ainda persistem no modelo TCP/IP com a possibilidade de exploração do *gap* temporal (tempo em segundos em que o *Controller front end* fica aguardando pela conexão TCP/443 do IP informado via SPA para iniciar a autenticação do usuário). Nesse tempo, ataques (e.g., *ip spoofing)* podem ocorrer entre o final do SPA e a conexão TCP subsequente. Um novo campo de *fingerprint* (i.e., resultado da construção da assinatura digital de um dispositivo computacional) é adicionado na estrutura do *payload* SPA. O método de manipulação (criação e uso) do novo campo também é apresentado. Usando atributos do dispositivo, do sistema operacional e dados do usuário, o método obtém um *fingerprint* robusto que dificulta falsificações.

Este capítulo propõe o desenvolvimento de uma solução que permite verificar se a conexão TCP, que sucede a autenticação do SPA, está sendo realizada pelo mesmo usuário e dispositivo que de fato enviou o SPA. Neste sentido, apresenta uma solução que altera o modelo de segurança de Perímetro Definido por Software, baseado em TCP/IP, para uma nova política baseada em dispositivos confiáveis. O escopo dessa proposta é genérico e pode ser adaptado a qualquer meio, dependendo do implementador da solução.

Os resultados experimentais mostram que a solução tem um baixo impacto no desempenho do SDP, dado que aumenta poucos milissegundos apenas na primeira fase da autenticação.

Este capítulo está organizado da seguinte forma: a seção 5.1 apresenta os trabalhos relacionados a *fingerprinting* de dispositivos; a seção 5.2 descreve o novo modelo de dados com *fingerprinting* de dispositivos; a seção 5.3 descreve o processo de *fingerprinting* de dispositivos; a seção 5.4 traz um método para inventário de dispositivos; a seção 5.5 apresenta o modelo de autenticação multi-factor proposto; a seção 5.6 descreve a autenticação de usuários com *fingerprinting* de dispositivos; a seção 5.7 apresenta as mudanças na arquitetura da conexão TCP subsequente; a seção 5.8

mostra os experimentos e resultados e, por fim, a seção 5.9 traz a conclusão parcial.

5.1 Trabalhos Relacionados a *Fingerprinting* de Dispositivos

Villela (2007) propôs um método para identificar dispositivos e controlar o acesso a um serviço. É realizado a coleta de dados relacionados a configurações de *software* e *hardware* de um dispositivo através de um agente de *software*, que gera uma assinatura digital e envia a um servidor de autenticação. Os *hashes* usados para gerar a assinatura digital são alterados a cada tentativa ao acessar um serviço. Em particular, o método descrito é aplicado ao acesso às informações confidenciais, contas bancárias *online*, transações comerciais (*e-commerce*) e informações sigilosas corporativas.

O trabalho de Booth e Kumhyr (2007) se refere ao controle e segurança de inventário de *hardware* de rede, o qual trata de um sistema para rastrear dispositivos (e.g., *laptops, smartphones*) perdidos ou roubados na *Internet*. Uma ferramenta de rastreamento de *hardware* montada na rede detecta as impressões digitais únicas do *hardware*. À medida que os dados passam por servidores que contêm a ferramenta de rastreamento, indícios de identificadores exclusivos (*hardware fingerprint*) como o endereço *Media Access Control* (MAC) são identificados. Os *fingerprints* digitais extraídos são, então, comparados com os *fingerprints* armazenadas em uma base de dados. Usando uma função de *hashing* ou mapeamento, o sistema do servidor é alertado para uma correspondência.

Uma identificação de dispositivos globalmente exclusiva é criada no trabalho de Rowland et al. (2008). Inicialmente, para um dispositivo, um processo é executado para obter informações sobre determinados componentes de *hardware,* como discos rígidos, placas de rede, som e vídeo, etc. Um identificador é atribuído ao dispositivo e armazenados em um repositório.

Fabricantes, vendedores ou licenciadores de computadores ou *hardware* de comunicação podem, às vezes, desejar restringir ou rastrear o uso de tais dispositivos ou *hardware* após a venda ou licença para outra parte. Etchegoyen (2014) propôs atribuir um identificador exclusivo ao componente de *hardware*. Gera uma *baseline*[34] do *fingerprint* para os componentes de *hardware* como, por exemplo, números de série, números de versão, datas e outros dados de *hardware, software* ou *firmware* instalados em um ou mais componentes de *hardware*. A resposta a uma consulta compreende ao identificador atribuído indicar se o *fingerprint* da *baseline* armazenado é idêntico ao gerado a um segundo *fingerprint* a partir do componente de *hardware.*

Outra técnica de *fingerprinting* abordada por Spear et al. (2016), Osborn et al. (2016) envolve a criação de um certificado privado com aspectos da configuração de *hardware* e *software* do dispositivo associado ao usuário. Pelo serviço de inventário,

[34] É uma “imagem” com as características atuais de um componente de *hardware* e funciona como um padrão oficial básico para as alterações subseqüentes.

os dispositivos terão as informações monitoradas em tempo real durante o acesso aos recursos da Organização. *Desktops* e *laptops* usam um certificado de máquina X.509[35] e uma chave privada. A limitação dessa técnica é o uso, apenas, em *desktops* e *laptops.*

Gardner e Volodarets (2017) propuseram uma ferramenta que fornece a capacidade para determinar a identidade de um dispositivo eletrônico, que executa um aplicativo de coleta dos principais atributos na forma de pontos de dados. O outro aplicativo de identificação está localizado no servidor remoto, o qual usa os principais pontos de dados para vincular o dispositivo a um proprietário ou entidade específica. A ferramenta rastreia as alterações nos principais pontos de dados associados ao dispositivo que, incluem o *hardware* e/ou *software* implementados no dispositivo eletrônico (e.g., marca e/ou modelo, número de série da placa mãe e do disco rígido, UUID, MAC *address*, número serial da memória RAM e bateria).

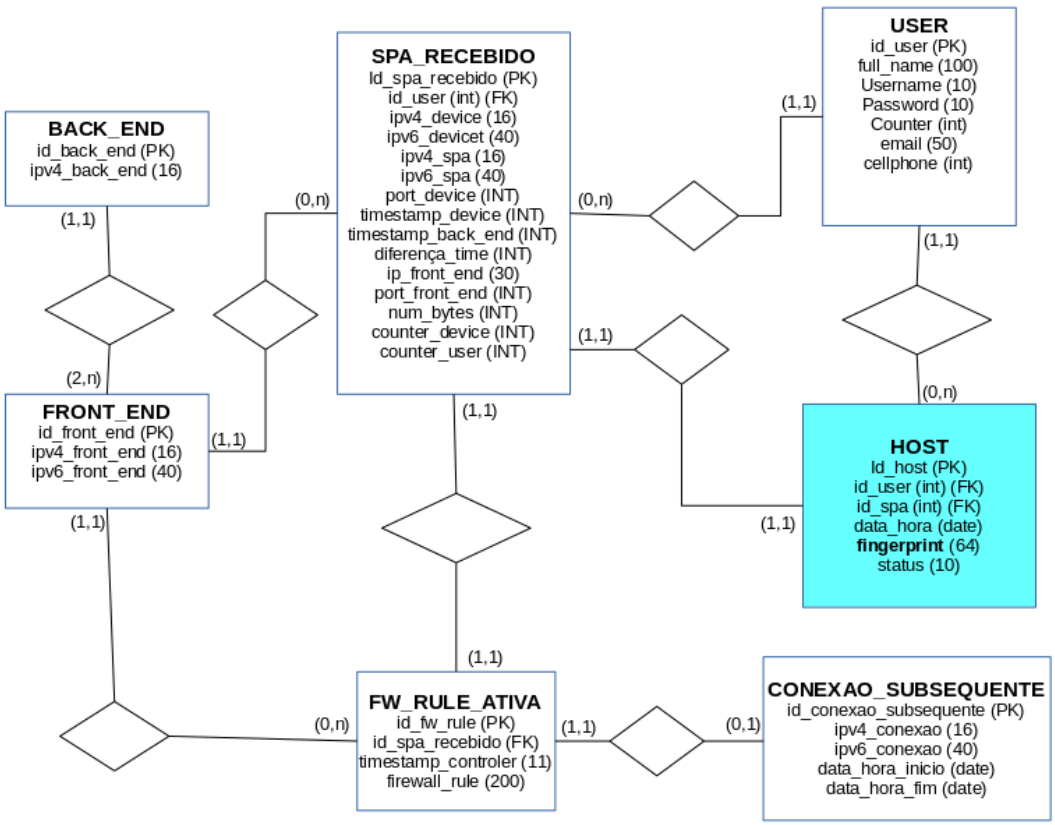
Com base nos trabalhos apresentados, essa pesquisa explora o uso de inventário de dispositivos computacionais para obter um *fingerprint* digital. O objetivo é mitigar ataques na arquitetura funcional do SDP, onde cada dispositivo passa a ser identificado de forma única para posteriormente ter permitido seu ingresso no perímetro. Com isso, um método para inventário de *desktops* e *laptops* é criado.

5.2 Novo Modelo de Dados com *Fingerprinting* de Dispositivos

O Modelo Entidade e Relacionamento apresentado anteriormente na Figura 5 (seção 4.2.2) precisou passar por alterações para suportar o *fingerprinting* de dispositivos. Com isso, a entidade “HOST” foi adicionada ao modelo de dados, visualizado na Figura 14 .

[35] Disponível em https://en.wikipedia.org/wiki/X.509

Figura 14 – Modelo ER da nova arquitetura SDP com a Entidade *HOST,* incorporando *fingerprinting* de dispositivos ao modelo.



Fonte: o Autor.

Um dispositivo pode ser caracterizado como uma coleção de componentes físicos ou virtuais que atuam como um computador, enquanto *host* é um *snapshot* (estado atual) do dispositivo em um dado ponto no tempo (e.g., um dispositivo pode ser um *laptop*, enquanto um *host* pode ser o sistema operacional e os dados coletados sobre o dispositivo).

Um usuário pode ter vários *hosts* de diferentes dispositivos ou um mesmo dispositivo pode possuir diversos *hosts*, devido a alterações ao longo do tempo ou possuir mais de um sistema operacional instalado nele. Não existe uma definição ou separação clara entre *host* e dispositivo, eles podem ter sentidos semelhantes e deve-se observar o contexto das colocações. No decorrer do trabalho, muitas vezes um dispositivo se refere a um *host.*

Um *HOST* sempre está associado a um usuário e a um SPA recebido. Ele possui os atributos *data_hora* (do seu recebimento), o *fingerprint* (*hash* de 32 *bytes*) e o *status*[36] do atributo *fingerprint*.

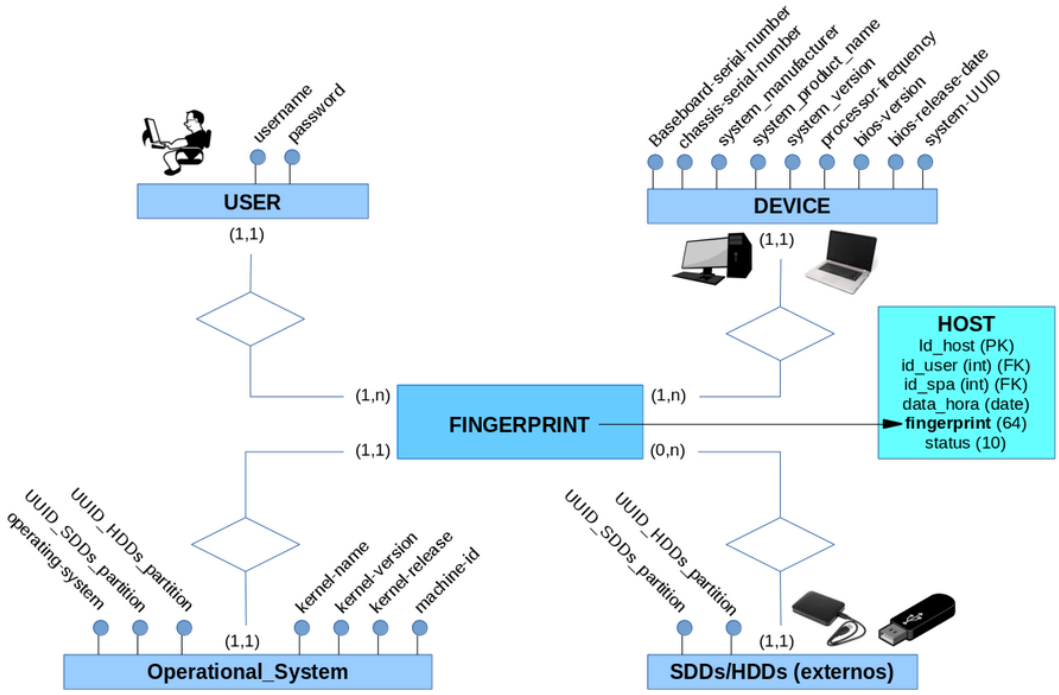
[36] Pode ser: novo, validado ou revogado.

5.3 Processo de *Fingerprinting* de Dispositivos

A falta de associação entre o processo do envio do SPA e a conexão TCP subsequente é um problema que ainda persiste em ambas técnicas de autenticação (SPA ou *port knocking*). Esse problema permite que um invasor se conecte a um servidor protegido em nome de um Cliente válido logo após o Cliente ter autenticado com sucesso no *Controller front end* e, antes de estabelecer uma conexão TCP com o servidor. A adoção de *fingerprint* do dispositivo Cliente no processo de autenticação pode mitigar essa vulnerabilidade.

Nessa seção, é proposto um método para montar um *fingerprint* a partir de atributos que apresentem relevância na identificação única do dispositivo. Os atributos foram escolhidos tendo como base os trabalhos relacionados e identificadores únicos (e.g., *username, password, serial-number, version, UUID*, etc). O método substitui a confiança em redes ou IPs por um nível apropriado de confiança no dispositivo. Cada dispositivo passa a ter um identificador consistente e não-clonável, enquanto as informações sobre o *software*, *hardware* e usuários devem ser integradas em um banco de dados de inventário, onde são representados por um *fingerprint* digital. A Figura 15 apresenta através de um mecanismo de abstração (entidades e atributos), a modelagem conceitual de criação do *fingerprint.*

Figura 15 – Modelagem conceitual de criação do fin*gerprint*.



Fonte: o Autor.

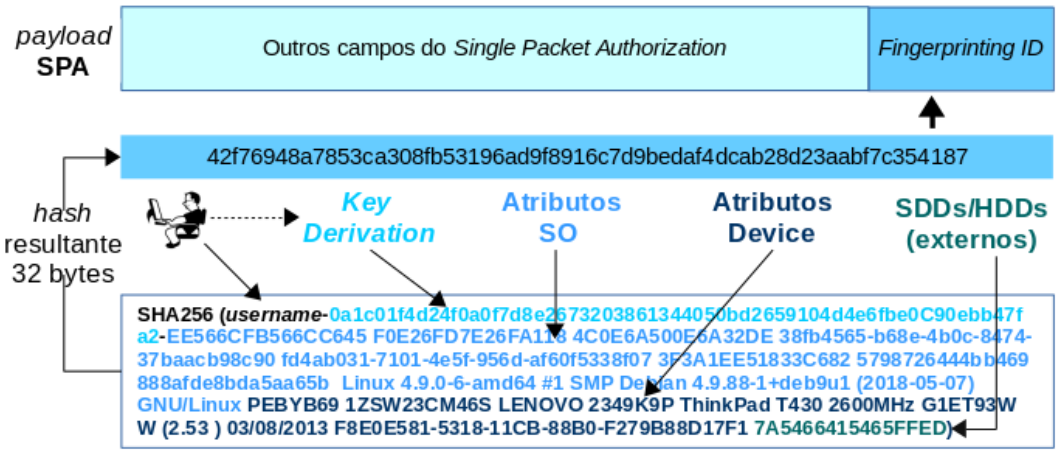
O *fingerprint* proposto é composto pelas seguintes entidades e atributos:

- *User*: é formado por dois atributos (*username* e *password*);

- *Device*: é uma coleção de componentes físicos ou virtuais. Foram escolhidos nove atributos, provavelmente únicos;
- *Operational_System*: em relação aos requisitos do sistema operacional. Foram selecionados sete atributos;
- SDDs/HDDs (externos): o usuário tem a possibilidade de usar, a seu critério, dispositivos externos (SSHDs/HDDs portáteis) geralmente conectados, via porta USB, para com os outros atributos formarem o *fingerprint* do dispositivo. Essa funcionalidade é similar a um *token* de *hardware*;
- *Fingerprint*: é a união de todos os atributos de todas as entidades aplicados a função de *hash* SHA-256.

Para detalhamento do processo de *fingerprinting*, apresenta-se a Figura 16. Com a seleção dos dados de entrada, o resultado da função SHA-256 retorna um *fingerprint* de 32 *bytes* que será incorporado ao *payload* do SPA para ser criptografado com os outros campos.

Figura 16 – Processo do *fingerprinting* e integração do *hash* com o SPA.



Fonte: o Autor.

As entidades representadas na modelagem conceitual de criação do *fingerprint* (Figura 15) possuem características que podem ser alteradas ao longo do tempo. A base de dados do *Controller back end* é atualizada se houver alguma mudança no dispositivo, informações do sistema operacional ou dados do usuário (e.g., uma partição adicionada ou deletada, *BIOS* atualizada em sua versão, usuário alterou sua senha, etc.). A alteração se dá quando o *fingerprint,* gerado pelo Cliente, é reconhecido como “novo” pelo *Controller back end*.

Para fortalecer o processo de *fingeprinting* é usada a derivação de chaves (seção 2.4). Com o uso da função PBKDF2, a senha mestra se torna uma chave derivada e alongada para posteriormente fazer parte do *fingerprint*. Foram usadas 1500 iterações

para derivação das chaves, pois número originalmente recomendado de iterações pela função PBKDF2 é 1000. No entanto este valor pode ser alterado conforme o poder de processamento de cada ambiente, pois é configurável. O tempo de processamento das iterações pode ser definido pelo administrador do sistema, tendo em vista elevar custos de ataques de força bruta ou dicionário em uma arquitetura cliente e servidor. Deve-se avaliar o tempo desejável e aceitável que irá impactar na geração do *fingerprint*.

A derivação de chaves é apresentada conforme a função:

$$Key3^{(1500)} = (HmacSha256(password \parallel salt \parallel 32))$$

Também se deve salientar que o número de iterações e o *salt* (salto criptográfico) são responsáveis pela diferenciação entre as chaves *Key1, Key2 e Key3.*

Por exemplo, a senha “1234” torna-se uma senha derivada e alongada de 32 *bytes*:

Key3=0a1c01f4d24f0a0f7d8e2673203861344050bd2659104d4e6fbe0c90ebb47fa2

5.4 Método para Inventário de Dispositivos

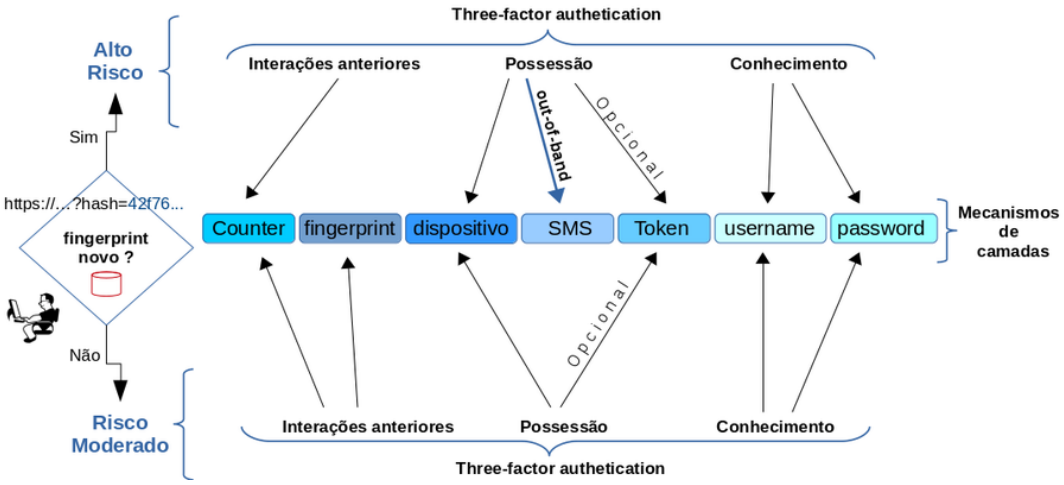
Para a obtenção do *fingerprint* digital, é necessário coletar informações sobre o dispositivo ou realizar seu inventário. Normalmente ele é realizado por um agente de *software* (VILLELA, 2007), um processo (ROWLAND et al., 2008), serviço de inventário (SPEAR et al., 2016) ou aplicativo de coleta de atributos (GARDNER; VOLODARETS, 2017). Todos possuem funcionalidades semelhantes, são instalados no dispositivo Cliente, coletam informações sobre dispositivos e geram um identificador exclusivo.

Este trabalho cria um aplicativo para realizar inventário de dispositivos (*desktops* e *laptops*) para obter um *fingerprint* digital. O usuário que deseja ingressar no perímetro deve fazer o *download* do aplicativo através de um canal *out-of-band* e instalá-lo.

5.5 Modelo de Autenticação Multi-factor

Na Figura 17 é apresentado o organograma do modelo de autenticação proposto por este trabalho. O modelo segue o modelo de autenticação baseada em riscos com múltiplos fatores.

Figura 17 – Autenticação *Multi-Factor* baseada em riscos.



Fonte: o Autor.

De acordo com a Figura 17, o modelo de autenticação *multi-factor* proposto explora a combinação de três fatores de autenticação: conhecimento, possessão e interações anteriores combinados com dois fatores de risco (alto e moderado). Todo acesso deve depender unicamente da identidade do dispositivo e das credenciais do usuário. Os acessos dos usuários não devem depender da rede de dados, seja ela dentro da empresa, uma rede doméstica, um aeroporto, hotel ou café, etc. Cada fator de autenticação abrange uma variedade de opções para autenticar o usuário ou dispositivo antes de prover acesso aos sistemas requisitados pelo mesmo.

Os acessos foram divididos em novo acesso (risco alto) e acesso de dispositivo já validado (risco moderado):

- **Novo acesso:** quando o *fingerprint* (*hash* de 32 *bytes*) enviado anteriormente via SPA for considerado “novo” (nunca recebido e não encontrado na base de dados) durante a requisição *https* ao *Controller back end,* o risco é classificado como alto. Ele precisa solicitar um fator de autenticação adicional (*nonce* enviado via SMS ao aparelho celular do usuário) através de um canal *out-of-band.* Isso limita a capacidade de um invasor usar credenciais roubadas;
- **Acesso de dispositivo já validado (demais acessos):** o risco moderado é caracterizado quando o *fingerprint* se encontra na base de dados do *Controller back end,* já validado anteriormente através de um *nonce* SMS enviado por um canal *out-of-band* ao aparelho celular pré-cadastrado do usuário.

Quando o risco é considerado **alto**, tem-se a abordagem de três fatores de autenticação e seus mecanismos:

1) Conhecimento: *username* e *password*;

2) Possessão: dispositivo, telefone celular (SMS), *token* (*pendrive* ou *hd* externo). O uso ou não de um *token* fica a critério do usuário;

3) Interações anteriores: valor da variável *Counter*.

Com o risco **moderado**, também, aborda-se três fatores de autenticação:

1) Conhecimento: *username* e *password*;

2) Possessão: dispositivo, *token* (*pendrive* ou *hd* externo). O uso ou não de um *token* novamente fica a critério do usuário;

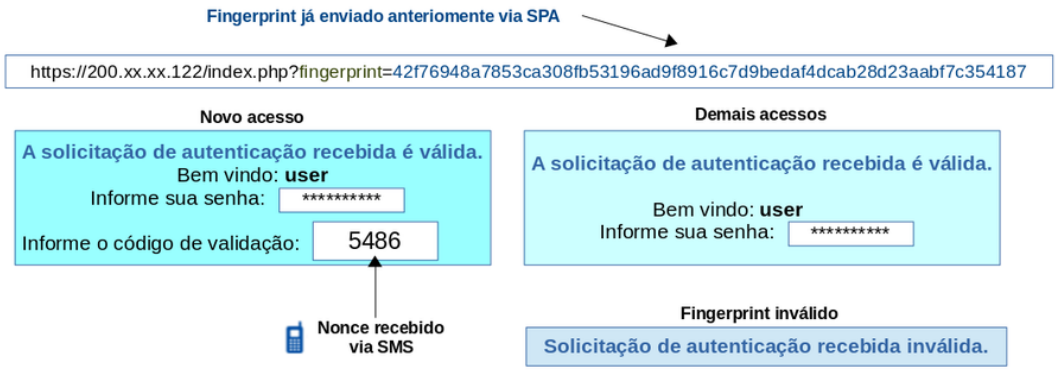
3) Interações anteriores: valor da variável *Counter* e *fingerprint* já validado e salvo na base de dados.

A autenticação *multi-factor* é uma prática que adiciona camadas (extras) de proteção ao nome de usuário e senha. Com a autenticação *multi-factor*, o usuário deve informar seu nome de usuário e senha como primeiro fator (o que ele sabe); como segundo fator (o que ele tem), através do seu dispositivo é coletado o *fingerprint* e opcionalmente do *token* de *hardware,* o seu UUID; e como terceiro fator (eventos passados de interação com o sistema) tem-se o valor da variável *Counter* ou *fingerprint*. Juntos, esses fatores fornecem maior segurança para ingresso em um SDP.

5.6 Autenticação de Usuários com *Fingerprinting* de Dispositivos

Após o envio do SPA, o aplicativo Cliente automaticamente inicia o navegador *web* e realiza uma conexão TLS. Nesse instante é enviado o *fingerprint* do dispositivo em sua requisição *https* ao *Controller front end* para validação. A Figura 18 mostra três possíveis respostas a requisição de autenticação.

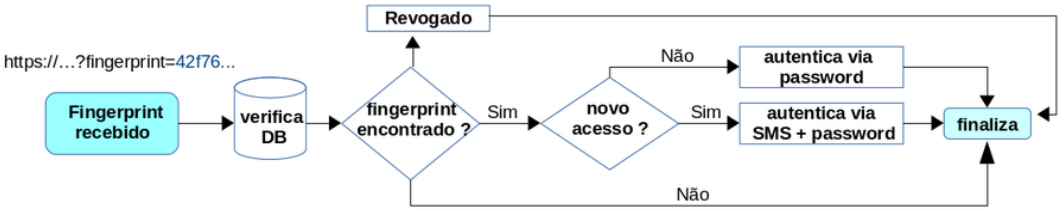
Figura 18 – Em um novo acesso o usuário deve digitar o *nonce* recebido pelo canal *out-of-band* (SMS). Nos demais acessos, apenas a senha é necessária.



Fonte: o Autor.

A cada conexão *https* ao *Controler front end*, o *hash* (*fingerprint* do dispositivo*)* da solicitação é verificado na base de dados. Em um novo acesso (quando for o primeiro acesso ou ocorrer alguma alteração[37] no *fingerprint* do dispositivo), um *nonce*[38] de 4 dígitos é enviado ao usuário via SMS. Todas as requisições de acesso ao perímetro devem pertencer a usuários já cadastrados e com número de contato via telefone celular válido. Nos demais acessos, quando o *Controller back end* recebe o *fingerprint* e verifica que seu *status* é *"validado"*, somente a senha de usuário é necessária. No caso de *fingerprint* inválido, uma mensagem de falha é retornada ao usuário. Por intermédio do canal *out-of-band* é possível revogar o *fingeprint* de um dispositivo. A Figura 19 apresenta o fluxograma do processo de autenticação de usuários via conexão *https.*

Figura 19 – Processo de validação do *fingerprint* de um dispositivo efetuado pelo *Controller front end.*



Fonte: o Autor.

Ao receber uma requisição *https* com o *fingerprint* (enviado anteriormente via SPA) em sua *url*, o *Controller front end* verifica na base de dados se o valor da variável *status* do *fingerprint*, que pode estar atribuída como:

- **novo:** a abordagem de alto risco (novo acesso) é considerada;
- **validado:** *fingerprint* já utilizado em acesso anterior, a abordagem de risco moderado é considerada;
- ***revogado:*** o dispositivo não tem permissão de acesso ao perímetro.

Uma das formas mais comuns de autenticar o usuário é através de solicitação de senha. Para uma autenticação mais forte, é usado o dispositivo e seu *fingerprint* (*hash* de 32 *bytes*) associado. O *nonce* de 4 dígitos enviados ao usuário via SMS e o valor da variável *status* fortalecem o modelo na totalidade. Com isso, tem-se a certeza de que: o usuário, o dispositivo e o endereço IP possuem credenciais verdadeiras para acesso ao perímetro.

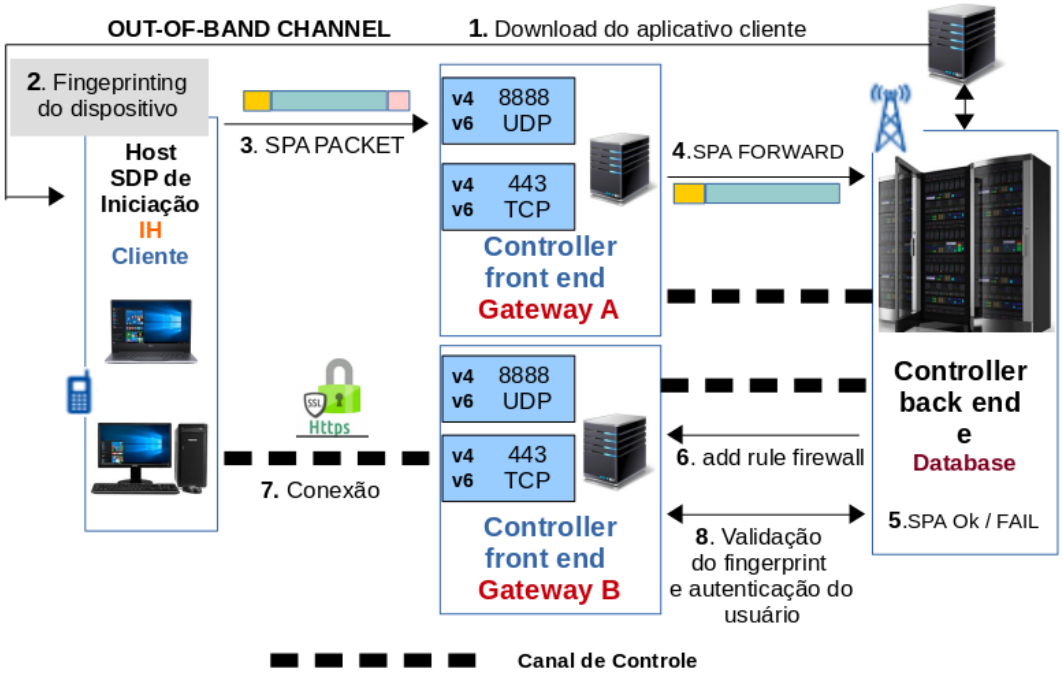
[37] Devido alguma alteração no *hardware*, sistema operacional ou dados do usuário.
[38] A validade do nonce recebido pelo Cliente possui uma validade customizável. Para esse trabalho foi definido um tempo máximo de 5 minutos.

5.7 Mudanças na Arquitetura da Conexão TCP Subsequente

Com a adoção do processo de *fingerprinting,* a arquitetura SPD apresenta novamente mudanças na estrutura.

Os *Controllers* devem ser identificados pelos seus endereços IP, nunca deve haver resoluções de nomes, evitando assim, ataques de DNS. A Figura 20 apresenta as alterações no projeto desenvolvido em relação à Figura 10 da seção (4.4).

Figura 20 – Arquitetura proposta com *fingerprinting* do dispositivo, envio do SPA, estabelecimento da conexão TCP subsequente e autenticação do Cliente.



Fonte: o Autor.

Com essa arquitetura, as políticas de acesso são baseadas em informações sobre um dispositivo e seu usuário associado. A seguir, a descrição das etapas:

1) *Download* do aplicativo (para obtenção do *fingerprint* e envio do SPA) via canal *out-of-band*;

2) *Fingerprinting* do dispositivo: ocorre criação do *fingerprint* para ser incluído ao *payload* do SPA;

3) SPA *Packet*: o SPA é criado e enviado a porta destino 8888 (UDP);

4) SPA (*Forward*): o *Controller front end* implementa condições de recebimento de um SPA. Após o recebimento, verifica-se se o *payload* contém os atributos previstos. Caso sim, o SPA é encaminhado ao *Controller back end* para validação;

5) SPA OK/*Failure*: o *Controller back end* recebe o SPA para processamento. Se as condições de validação forem satisfatórias, os dados do SPA são salvos na base de dados, caso contrário ele é descartado;

6) ADD *Rule Firewall:* liberação do IP recebido via SPA e porta TCP/443 no *Controller Front end*;

7) Conexão *https*: alguns segundos, após o envio do SPA, o dispositivo Cliente já pode estabelecer o TLS mútuo. Para isso, o *Controller front end* aguarda por um tempo configurável pelo *TCP SYN PACKET* do Cliente para iniciar o *handshake* TLS. O próprio aplicativo Cliente inicia o navegador *web* padrão já contendo em sua *url,* o IP do *Controller front end* e o *fingerprint* na requisição, já enviado anteriormente via SPA;

8) Nessa fase ocorre a validação do *fingerprint.* Caso a validação obtenha sucesso, é iniciado a autenticação do usuário.

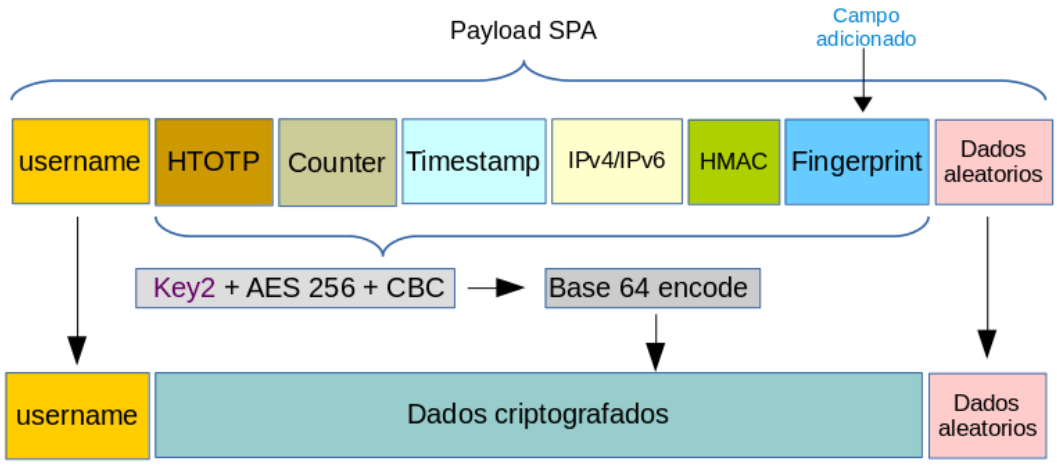
Para cada novo *fingerprint* recebido via SPA é gerado um *nonce* via *Pseudo-random Number Generator* (PRNG) de 4 dígitos, que será enviado posteriormente via SMS ao usuário como segundo fator de autenticação. O valor do *nonce* é calculado como base no total em segundos passados desde 1.º de janeiro de 1970 até a data/hora atual. Dessa forma, a cada execução, o valor da "semente" será diferente.

No sistema existem diversos parâmetros que devem ser configurados pelo usuário e pelo administrador do sistema. Os parâmetros configurados pelo usuário são realizados dentro do aplicativo cliente, onde ele informa o seu *username* e *password*. Já o administrador do sistema deve configurar o valor de iterações das chaves *Key1, Key2 e Key3* e salto criptográfico, porta UDP do *Controller front end* para recebimento do SPA, valor inicial do *Counter* (contador) sincronizado entre o Cliente e o *Controller back end* e por fim, as unidades de tempo anterior e posterior ao tempo atual do *Controller back end* que o pacote SPA pode ser aceito.

5.8 Experimentos e Resultados

Essa seção apresenta os resultados obtidos por meio da avaliação da abordagem proposta. A Figura 21 identifica o campo *fingerprint* adicionado ao *payload* SPA para ser criptografado e enviado ao *Controller front end.*

Figura 21 – Campo *fingerprint* adicionado ao *payload* SPA.



Fonte: o Autor.

Na Figura é observado o resultado da construção da assinatura digital de um dispositivo computacional e como ele é adicionado na estrutura de *payload* do SPA através do campo *fingerprint*. Posteriormente ele é criptografado e enviado ao *Controller front end.* A seguir são apresentados experimentos realizados a partir de testes quantitativos.

5.8.1 Análise Quantitativa

Nos experimentos quantitativos foram medidos os tempos de processamento em milissegundos à execução de aplicações de testes. As aplicações foram desenvolvidas na linguagem de programação ANSI C. Os atributos para formação do *fingerprint* foram coletados com as ferramentas *dmidecode*[39], *uname*[40] e *blkid*[41].

De modo a auxiliar na obtenção dos valores quantitativos se desenvolveu uma função, que realiza pontos de marcação do tempo, no início e fim do processamento no módulo *key derivation, fingerprinting* e SPA. Foram realizadas 30 repetições para cada conjunto de testes.

A Tabela 5 mostra os resultados obtidos durante a criação do *fingerprint*, a geração da *key derivation* e criação/recebimento do SPA. A primeira observação diz que o uso de *key derivation* possui um tempo de processamento de 3,92 ms tendo em vista as 1500 iterações (valor customizável). A senha mestra se torna uma chave derivada e alongada antes de ser usada pelo processo de *fingerprinting*.

O número de iterações a ser usado está ligado diretamente ao custo de processamento e memória, pois a ideia é elevar custos de um ataque de força bruta ou

[39] Disponível em: https://linux.die.net/man/8/dmidecode
[40] Disponível em: https://linux.die.net/man/1/uname
[41] Disponível em: https://linux.die.net/man/8/blkid

dicionário. Aumentar o valor das iterações inevitavelmente aumenta o custo de geração do *fingerprint*.

Tabela 5 – Resultados da simulação quantitativa em milissegundos.

Fatores de Processamento	Cliente	*Controller back end*
Key Derivation (1500 iterações)	3,92	não considerado
Fingerprinting do dispositivo	10,23	não considerado
Criação/recebimento do SPA	10,43	14,58
Total	**24,58**	**14,58**

Fonte: o Autor.

A segunda observação é um acréscimo de 10,23 ms durante o processo de formação do *fingerprint* pelo Cliente. O processo envolve a leitura de todos os atributos e geração do *hash* resultante de 32 *bytes* pela função SHA-256. O processamento é insignificante, pois é efetuado somente uma vez no início de uma conexão com o SDP.

A terceira observação é um tempo de 10,43 ms na criação e 14,58 ms no processamento de recebimento do SPA. Com o *fingerprint* incorporado ao SPA, houve um aumento de 1,06 ms no Cliente e 1,14 ms no *Controller back end* em relação aos resultados anteriores do processamento do SPA (Tabela 3).

A quarta observação é o tempo total de processamento, no dispositivo Cliente foi de 24,58 ms, e no *Contoller back end* foi de 14,58 ms.

Através das observações feitas, concluí-se que a incorporação do processo de *fingerprinting* de dispositivos ao SPA aumenta os níveis de segurança na autenticação com um baixíssimo custo computacional.

5.9 Conclusão Parcial

Este capítulo demonstrou que o problema relacionado à identificação IP, que pode ser explorado no *gap* entre a autenticação do SPA e estabelecimento da conexão subsequente, pode ser mitigado pela adoção de *fingerprinting* de dispositivos.

Com a implementação do processo de *fingerprinting*, obtêm-se informações consistentes e únicas sobre cada dispositivo. Os resultados mostram, por meio da análise quantitativa, que a adição do tempo de processamento de *fingerprinting* é mínima e se concentra no lado do Cliente.

Um mecanismo interessante na solução é a possibilidade de utilizar dispositivos externos (SSHDs/HDDs portáteis) para a formação do *fingerprint*. Diante dessa funcionalidade, o usuário tem a possibilidade, a seu critério, em usar seus dispositivos externos para formar o *fingerprint*, remetendo a ideia de um *token.*

Os resultados obtidos são bastante significativos e apontam para novas possibilidades em como usuários substituem as políticas baseadas em TCP/IP pela confiança no seu dispositivo, corrigindo falhas do modelo.

6 VALIDAÇÃO DA SEGURANÇA

A *Internet* foi, inicialmente, projetada sem muitas considerações de segurança e vem evoluindo há anos com correções para resolver várias falhas em sua estrutura (PETERSON; DAVIE, 2013). Ataques costumam ocorrer na *Internet* com diversos objetivos, visando diferentes alvos e usando variadas técnicas. Qualquer serviço, computador ou rede que seja acessível via rede mundial de computadores pode ser alvo de ataque, assim como qualquer computador com acesso à *Internet* pode participar de um ataque.

Uma vulnerabilidade é definida como uma condição que, explorada por um atacante, pode resultar em uma violação de segurança (CERT.BR, 2017/03/16). Exemplos de vulnerabilidades são falhas no projeto, na implementação ou na configuração de programas, serviços ou equipamentos de rede. Um ataque de exploração de vulnerabilidades ocorre quando um atacante, utilizando-se de uma vulnerabilidade, tenta executar ações maliciosas, como invadir um sistema, acessar informações confidenciais, disparar ataques contra outros computadores ou tornar um serviço inacessível (CERT.BR, 2017/03/16).

Para validação, desse trabalho, foram escolhidos 20 ataques a partir de publicações prévias que envolvam defesa de perímetro (PUTHAL et al., 2017; LIEW et al., 2010; KUMAR; TALWAR, 2012; PETERSON; DAVIE, 2013; ZORKTA; ALMUTLAQ, 2012; TARIQ; BAIG; SAEED, 2008). A principal característica dos ataques a serem selecionados foi a possibilidade de que ocorram em uma arquitetura cliente e/ou servidor. Considerando o modelo TCP/IP de encapsulamento, os ataques definidos atuam em uma ou mais camadas (e.g., IP, transporte ou aplicação). Na atualidade, existem centenas de ataques na *Internet* variando da simples diversão até a realização de ações criminosas.

A validação da abordagem proposta é baseada na descrição dos ataques e de como são mitigados, tendo em vista os recursos descritos e implementados por esse trabalho.

1. *Replay attack*

Descrição: geralmente é descrito como uma sequência de mensagens retransmitidas entre as partes da comunicação. A repetição envolve a captura de uma unidade de dados e sua subsequente retransmissão para obter um efeito não autorizado. Por exemplo, um atacante escuta e captura sequências de interações de autenticação e pode reutilizar a mensagem para enganar o autenticador. O invasor pode usar a mensagem ou cópias, da mesma, para iniciar ataques (LI, 2009).

Mitigação: além do uso do contador baseado em evento proposto pelo protocolo SDP v1.0, esse trabalho usa o *timestamp* e a senha derivada *Key1*, formando assim um *nonce* sólido, evitando ataques de repetição e fortalecendo a originalidade do SPA. O *nonce* é individualizado por usuário e validado pela nova função HTOTP criada.

2. *Dictionary e brute-force password search attack*

Descrição: os esquemas tradicionais de senha baseados em texto estão sujeitos a ataques de dicionário em uma escala muito grande. Ataques de dicionário têm uma alta eficiência se a senha secreta for uma palavra comum. Um ataque de dicionário é limitado a correspondências exatas, mas ainda é surpreendentemente bem-sucedido, pois os usuários tendem a escolher senhas simples e previsíveis. Os ataques de dicionário, com exceção de abordagens híbridas (combinações de palavras entre diferentes dicionários) são muito mais rápidos do que o método básico de força bruta (IRFAN et al., 2018)**.** Já o ataque de força bruta tenta procurar uma palavra em um banco de dados de senhas tentando todas as combinações de letras, números e símbolos possíveis (GAUTAM; JAIN, 2015);

Mitigação: se um atacante interceptar um SPA e descobrir a senha do usuário, ele pode recriar o SPA e ingressar no perímetro. O uso de derivação de chaves torna mais difícil a quebra de senhas. A *Key1* foi usada para os módulo originalidade, autenticidade e integridade de dados; a *Key2* é usada na criptografia do *payload* e a *Key3* é aplicada ao *fingerprinting* do dispositivo. Além disso, o número de interações pode ser aumentado com o objetivo de elevar custos de ataques, inevitavelmente aumenta os custos da autenticação do usuário legítimo. Por ser configurável, cada administrador do sistema deve definir o tempo de processamento (iterações) que irá impactar na autenticação. O uso do *fingerprint* também ajuda a evitar estas categorias de ataques, pois o *hash* enviado previamente via SPA habilita por um tempo configurável o *Controller front end* a somente responder a requisição de um *fingerprint* válido.

3. Packet crafting attack

Descrição: *crafting* significa fazer ou criar alguma coisa habilmente. A criação de pacotes também não é uma exceção e, acredita-se que seja um tipo tecnicamente avançado de exploração de vulnerabilidades, que dificulta a detecção e o diagnóstico. *Packet crafting* vai além, tentando testar a presença, a funcionalidade ou a precisão das regras de *firewall* e dos sistemas de detecção de invasão. A criação de pacotes requer um conhecimento profundo do protocolo TCP e, seu funcionamento é mais um ataque orquestrado manualmente do que um automatizado. Isso faz com que seja uma maneira tecnicamente avançada de tentar invadir redes e sistemas. Na elaboração de pacotes, cria-se um completamente novo ou se edita o existente para alterar as informações. Então, é enviado para ver a resposta do *firewall* ao alterar os valores. Com isso, os invasores tentam encontrar o ponto de entrada de uma rede de computadores.

Mitigação: a criação do SPA requer passos bem definidos. Por exemplo, o atacante ao criar um pacote precisa das senhas derivadas (*Key1, Key2* e *Key3*) obtidas a partir da senha do usuário. O módulo de derivação de chaves e a criptografia do *payload* são

recursos que evitam a decodificação de um SPA e criação de um falso. Uma simples falha na criação de um dos módulos inviabiliza a execução dos outros. Nesse sentido, alterar ou criar um *payload* falso passa a ser uma tarefa árdua para os atacantes.

4. *Source IP spoofing attack*

Descrição: em redes TCP/IP, os pacotes enviados de um *host* para outro incluem um cabeçalho IP que contém, entre outros, endereço IP de origem e destino. O endereço IP de origem identifica o endereço IP do *host* de envio e o IP do destino, o de recebimento. O *host* destinatário direciona as respostas ao remetente usando esse endereço IP de origem. No entanto, o destinatário não tem meios para validar a autenticidade do endereço de origem do pacote. Essa vulnerabilidade pode ser explorada por invasores para enviar pacotes com endereço IP de origem alterado ou falsificado. O envio de endereço de origem forjado é conhecido como *spoofing* de pacote ou falsificação de IP de origem (NOURELDIEN; HUSSEIN, 2009).
Mitigação: o *fingerprint* enviado via SPA e a solicitação de validação do *fingerprint* via conexão TLS subsequente, evitam que um atacante ingresse no perímetro através de um ataque de IP *spoofing*. O *Controller front end* somente envia respostas a requisições que contenham um *fingerprint* válido. O endereço de IP de origem deixa de ter um papel único na identificação do usuário, ele continua fazendo parte do processo, mas em um aspecto secundário. Falsificar o IP continua trivial e, não é mais viável ingressar no perímetro apenas forjando um IP, pois o usuário precisaria conhecer e ter os fatores de autenticação usados nesse trabalho.

5. *Connection TCP hijacking attack*

Descrição: um invasor pode inserir pacotes falsificados em uma conexão TCP entre o cliente e o servidor, permitindo acesso ao servidor e falsificando mensagens que parecem vir do cliente. Todas as implementações do TCP são vulneráveis. O invasor cria pacotes sincronizados que imitam os que estão sendo transmitidos pelo cliente e pelo servidor, comprometendo a integridade e autenticidade da comunicação, modificando ou injetando mensagens adicionais (HAGEN; MULLINS, 2013);
Mitigação: o TCP é somente usado durante a conexão subsequente. Nesse sentido, o protocolo TLS estabelecido entre o Cliente e o *Controller front end* tende a evitar esse tipo de ataque. Outro fator que evita o comprometimento da integridade e autenticidade da comunicação é o *fingerprint* do dispositivo. Ao mesmo tempo, a autenticação *multi-factor* baseada em riscos abrange uma variedade de opções para proteger o processo de autenticação do dispositivo e do usuário.

6. *Stolen-verifier attack*

Descrição: o *stolen-verifier attack* é quando um adversário rouba o verificador de

senha do servidor e pode usá-lo diretamente para se passar por um usuário legítimo em um processo de autenticação. Se um adversário tiver roubado o esquema *password-verifier* do servidor, ele poderá montar um ataque de adivinhação. Usuários comuns, provavelmente escolherão uma senha simples para facilitar a memorização. O adversário pode adivinhar senhas por ataque de dicionário e verificar quando o *hash* é igual à base roubada (ZHU; YU; ZHANG, 2008). Se o verificador de senha for roubado, o invasor poderá atuar como servidor, porque tanto o servidor quanto o invasor tem a mesma quantidade de dados para a autenticação de clientes.
Mitigação: o que não pode ser visto não pode ser roubado. Roubar o verificador de senha do *Controller back end* é muito difícil devido à ocultação de toda infraestrutura proposta pelo protocolo SDP. Tanto a Especificação v1.0 quanto a nova arquitetura proposta, são imunes a este tipo de ataque.

7. *Spoofed packets attack*

Descrição: é um dispositivo conectado à *Internet* que está tentando se passar por um remetente legítimo e autêntico. São criados ao modificar as informações sobre o endereço de origem na seção de cabeçalho de um pacote IP. O *spoofing* é usado pelo *hacker* para ocultar sua identidade e presença na rede. Os pacotes falsificados ou replicados também são usados para derrotar o mecanismo de segurança de uma rede e ignorar os serviços baseados na autenticação IP. Além disso, outros campos do cabeçalho TCP podem ser alterados (e.g., número de sequência, *timestamp*, porta de origem ou destino).
Mitigação: os fatores de autenticação usados no trabalho identificam a entidade reivindicada, composta por um dispositivo e uma pessoa. A identificação do dispositivo Cliente, através do *fingerprint* (ao estabelecer a conexão TLS) e a autenticação *multi-factor* não permitem que outros dispositivos tentem ingressar no perímetro, fazendo-se passar por dispositivos que procuram ocultar sua identidade.

8. *Man-in-the-middle attack*

Descrição: é um ataque ativo realizado em tempo real. Ocorre quando um usuário deseja conectar dois dispositivos, mas em vez de se conectar uns com os outros, conectam-se inadvertidamente a um terceiro dispositivo. O terceiro dispositivo retransmite as informações entre os dois dispositivos. Nesse ataque, todas as informações enviadas entre os dispositivos são comprometidas.
Mitigação: um dos ataques mais poderosos da *Internet*. Todo e qualquer recurso de segurança é útil contra o ataque *man-in-the-middle*. Todos os recursos SDP descritos pela Especificação 1.0 e os novos recursos propostos são usados combater esse tipo de ataque. Salienta-se que a derivação de chaves, o *fingerprinting* do dispositivo, a nova arquitetura SDP, a conexão TLS e o SMS enviado por um canal *out-of-band* são

recursos que se destacam.

9. *Port scanning attack*

Descrição: são ferramentas com o objetivo de mapear as portas TCP e UDP e, geralmente apresentá-las em seis estados: aberto (*open*), fechado (*closed*), filtrado (*filtered*), não-filtrado (*unfiltered*), *open/filtered* ou *closed/filtered*. Comumente são usados por pessoas mal intencionadas para identificar portas abertas e realizar invasões.
Mitigação: o Protocolo SDP não permite este tipo de ataque devido à ocultação dos *Controllers* e recursos protegidos (servidores, *gateways*, redes, serviços, etc.). A arquitetura SDP, tanto a v1.0 quanto a proposta por este trabalho inibem este tipo de ataque.

10. *DNS spoof attack*

Descrição: este ataque altera os endereços IPs dos servidores DNS da vítima e apontam a servidores maliciosos. Existem diferentes formas de modificar os endereços IPs do DNS. O caso mais comum envolve usuários domésticos através de roteadores mal configurados em suas casas. O cibercriminoso pode estabelecer um servidor DNS falso e as buscas da vítima passarão por este servidor e, dessa forma, será possível redirecioná-lo para qualquer *site* malicioso.
Mitigação: conforme o especificado pelo módulo confidencialidade de tráfego, o *Controller front end* é sempre identificado pelo seu endereço IP e nunca há resolução DNS pelo aplicativo Cliente.

11. *DNS cache poisoning attack*

Descrição: também conhecido como envenenamento de *cache* DNS. Quando um servidor DNS recebe dados não autenticados e armazena os mesmos em sua *cache* para otimizar o desempenho, tais informações podem ser falsas e pode haver o envenenamento do servidor, que fornece os dados não autenticados para seus clientes. Se um servidor DNS é envenenado, esse pode retornar o endereço IP incorreto, desviando o tráfego para outro computador.
Mitigação: o envenenamento de *cache* de servidores DNS não caracteriza um risco para esse trabalho, pois nenhuma resolução DNS é realizada pelo aplicativo Cliente. Todos os *Controllers front end* são identificados pelos seus IPs, conforme definido pelo módulo de confidencialidade de tráfego.

12. DoS *e* DDoS *attack*

Descrição: um ataque de negação de serviço (DoS *Attack,* do inglês *Denial of Service*), é uma tentativa de tornar os recursos de um sistema indisponíveis para os seus usuários. Em um ataque distribuído de negação de serviço (DDoS, do inglês *Distributed*

Denial of Service), um computador mestre denominado *master* pode ter sob seu comando até milhares de computadores zumbis. O ataque consiste em fazer com que os zumbis (máquinas infectadas e sob comando do Mestre) se preparem para acessar a um determinado recurso, num determinado servidor, numa mesma hora e data tornando o acesso indisponível a quem de verdade deseja acessar o site ou serviço.

Mitigação: a arquitetura SDP torna *os controladores*, servidores, *gateways*, redes e serviços invisíveis na *Internet*. A concepção do SDP inclui à ocultação da infraestrutura, de serviços e controles de acesso. Somente dispositivos devidamente autorizados via SPA podem ingressar no perímetro. No caso do SDP, o atacante pode explorar a abertura da porta 443, o que levaria a negação de serviço ao Cliente que enviou o SPA e não a outros Clientes. Com a nova solução, nem mesmo este Cliente tem seu serviço negado, pois o serviço é provido por um *Controller* distinto do que autenticou o SPA, dificultando ainda mais a sua localização.

13. *Man-in-the-browser attack*

Descrição: o ataque *man-in-the-browser* (MitB) é um tipo *man-in-the-middle* (MitM). Técnicas de ataque do tipo *man-in-the-middle* estão, principalmente, direcionando o fluxo de informações entre um cliente e um servidor e agora evoluíram para se tornarem ataques *man-in-the-browser*. O ataque MitB é projetado para se infiltrar no *software* cliente, como o navegador da *Internet*, e manipular ou roubar informações confidenciais (NOR; JALIL; MANAN, 2012).

Mitigação: o *fingerprint* do dispositivo com à autenticação *multi-factor* tende a evitar este tipo de ataque. O adversário pode tentar capturar dados durante a etapa (7) da Figura 20. O protocolo TLS, o *nonce* SMS e o *fingerprint* atuam para barrar a interceptação.

14. 0*-day exploit attack*

Descrição: um ataque de *0-day* acontece quando uma falha, ou vulnerabilidade de *software/hardware* é explorada e os invasores liberam *malware,* antes que um desenvolvedor tenha a oportunidade de criar um *patch* para corrigir a vulnerabilidade/falha.

Mitigação: tanto a arquitetura SDP como a nova arquitetura proposta inibem esse tipo de ataque. No entanto, o *fingerprint* e a autenticação *multi-factor* são recursos que podem evitar que essa forma de ataque tenha sucesso numa invasão de perímetro caso o dispositivo Cliente for comprometido.

15. *Phishing attacks*

Descrição: é uma maneira desonesta que cibercriminosos usam para enganar usuários a revelar informações pessoais, como senhas do cartão de crédito, CPF, números

e senhas de contas bancárias, etc. O atacante faz isso enviando *e-mails* falsos ou direcionando a *web sites* enganosos. As principais vítimas desse ataque são os usuários e seus dispositivos.
Mitigação: mesmo que o usuário acesse um *Controller front end* falso, o *fingerprint* enviado anteriormente via SPA e o SMS, necessário para o primeiro acesso, podem barrar este tipo de ataque.

16. *Evesdropping attack*

Descrição: é a interceptação não autorizada, em tempo real, de uma comunicação privada como uma chamada telefônica, mensagem instantânea, videoconferência ou transmissão de dados. O termo e*vesdropping* (espionagem) deriva da prática de ficar de pé sob o beiral de uma casa, ouvindo conversas no interior.
Mitigação: caso o SPA seja interceptado, a derivação de chaves (*Key1, Key2* e *Key3*) podem impedir que um invasor descubra a senha mestra de entrada. Na hipótese da interceptação da conexão TCP subsequente, o *fingerprint* (*hash* de 32 *bytes*) embutido na requisição *https* e enviado anteriormente via SPA pode impedir esse tipo de ataque.

17. *Pharming attack*

Descrição: os ataques *pharming* podem ser conduzidos pela exploração de vulnerabilidades em servidores DNS ou arquivos de *hosts* de computador, usados pelo sistema operacional, para mapear nomes de *host* para endereços IPs. É uma prática fraudulenta de direcionar usuários da *Internet* para um *site* falso que imita a aparência de um *site* legítimo, de modo a obter informações pessoais, como senhas, números de conta, etc. Um ataque *pharming* ajudará os invasores a realizar seus cenários de ataque *phishing* de uma maneira mais sofisticada para torná-lo confiável e mais difícil de descobrir que o usuário está sendo atacado.
Mitigação: todos os *Controllers front end* são identificados por endereços IPs. O dispositivo Cliente não consulta servidores DNS. Caso o tráfego tenha sido redirecionado para um site falso, o SPA enviado anteriormente, o *fingerprint (hash* de 32 *bytes)* ligado ao dispositivo e a autenticação *multi-factor* provêm ações para barrar o ataque.

18. *Defacement attack*

Descrição: é uma técnica que consiste na realização de modificações de conteúdo e estética de uma página da *Web.* Geralmente ocorre por vulnerabilidades do servidor de aplicação e erros da aplicação *Web*. Esse tipo de ataque pode ter como alvo o *Controller front end.*
Mitigação: a arquitetura SDP e a nova arquitetura proposta tornam invisível qualquer dispositivo conectado a *Internet*, tornando o *Controller front end* imune a este tipo de

ataque.

19. *Rainbow attack*

Descrição: o invasor pode facilmente usar o *hash* pré-calculado e combinar como texto simples para quebrar o *hash* desejado. Esses ataques são iniciados pela construção de *rainbow tables* que armazenam um valor com *hash* para cada palavra no dicionário de chaves. Uma chave criptográfica pode ser quebrada procurando os valores de *hash* e sua chave de texto simples correspondente na *rainbow tables* (TAHIR et al., 2013).
Mitigação: caso o SPA for interceptado, o atacante tentará descriptografar o pacote usando *rainbow tables*. A derivação de chaves, um número de derivações pré-definido e um *salt* adicionado ao processo de formação da senha fortalece a proteção contra esse tipo de ataques reduzindo a capacidade de usar *hashes* pré-computados.

20. *Impersonation attack*

Descrição: é um ataque no qual um adversário assume com sucesso a identidade de uma das partes legítimas no sistema ou em um protocolo de comunicação. Tais ataques são, geralmente, direcionados a funcionários corporativos. O ataque é executado enviando um *e-mail* para o destino no qual o remetente tenta se mascarar como uma fonte confiável. Isso é feito para obter acesso às informações confidenciais, como dados financeiros.
Mitigação: a identificação das entidades reivindicadas é composta por um dispositivo e uma pessoa. Nesse sentido, a autenticação *multi-factor* e o *fingerprint* do dispositivo barram este ataque caso o usuário seja enganado e acesse um *Controller front end* falso na sua autenticação.

A sumarização desse trabalho é dada pela associação dos vinte ataques descritos com as seguintes colunas da Tabela 6:

- **Não SDP**: refere-se ao processo de autenticação do modelo tradicional de perímetro baseado em *firewall;*
- **SDP v1.0**: (1) Autenticidade, (2) Originalidade, (3) Arquitetura SDP.
- **SDP vUFSM**: (1) Autenticidade, (2) Originalidade, (3) Nova Arquitetura SDP, (4) Derivação de chaves, (5) Pontualidade, (6) Integridade, (7) Confidencialidade de Dados, (8) Confidencialidade de Tráfego, (9) *Fingerprinting* de dispositivos, (10) Autenticação *multi-factor*.

Tabela 6 – Comparativo entre os modelos de defesa de perímetro.

ID	Ataques	Não SDP	SDP v1.0	SDP vUFSM
1	*replay attack*	vulnerável	2	2
2	*dictionary e brute-force password search*	vulnerável	vulnerável	4 e 9
3	*packet crafting*	vulnerável	vulnerável	4 e 7
4	*source IP spoofing*	vulnerável	vulnerável	9 e 10
5	*connection TCP hijacking*	vulnerável	vulnerável	9 e 10
6	*stolen-verifier attack*	vulnerável	3	3
7	*spoofed packets*	vulnerável	vulnerável	9 e 10
8	*man-in-the-middle attack*	vulnerável	1, 2 e 3	1 ao 10
9	*port scanning*	vulnerável	3	3
10	*DNS spoof*	vulnerável	vulnerável	8
11	*DNS cache poisoning*	vulnerável	vulnerável	8
12	*DOS e DDOS attacks*	vulnerável	3	3
13	*man-in-the-browser*	vulnerável	vulnerável	9 e 10
14	*0-day exploit attack*	vulnerável	3	3, 9 e 10
15	*phishing attacks*	vulnerável	vulnerável	9 e 10
16	*evesdropping attack*	vulnerável	vulnerável	4 e 9
17	*pharming attacks*	vulnerável	vulnerável	8, 9 e 10
18	d*efacement attack*	vulnerável	3	3
19	*rainbow attack*	vulnerável	vulnerável	4
20	*Impersonation attack*	vulnerável	vulnerável	9 e 10

Fonte: o Autor.

6.1 Conclusão Parcial

Esse capítulo aborda a validação da solução proposta. Apresenta 20 ataques selecionados de publicações prévias que envolvam defesa de perímetro correlacionados a três abordagens: 1) o modelo tradicional de perímetro baseado em *firewall;* 2) o SDP descrito pela Especificação 1.0; 3) o novo SDP apresentado por esse trabalho.

O modelo tradicional de perímetro baseado em *firewall* apresenta um ambiente inseguro em relação ao quesito segurança. O *firewall* passa a falsa ideia de que ele é a solução dos problemas de segurança. Há pouco tempo era fácil definir um *firewall* e suas funções, pois o perímetro era facilmente definido. Assim, o *firewall* é fundamental, mas não é tudo. O enfoque da segurança, agora, está em selecionar os usuários que podem acessar a rede e definir os direitos que têm. Além de determinar os recursos que cada usuário, em particular, pode acessar e os níveis de acesso de cada um na rede (NAKAMURA; GEUS, 2007).

Já, o modelo de perímetro descrito pela Especificação 1.0 do SDP apresenta um desempenho de segurança satisfatório. Ele apresenta uma nova ideia para autenticar antes de a primeira comunicação acontecer. Observou-se que 35% dos ataques podem ser combatidos com esse modelo.

O novo modelo de acesso a perímetro baseado em SDP apresentado, por esse trabalho, apresenta eficiência na sua totalidade. O modelo segue os preceitos do SDP e acrescenta melhorias na concepção do SPA, resolve o problema do protocolo TCP/IP e cria uma abordagem de recebimento do SPA e da conexão subsequente.

7 CONCLUSÕES FINAIS

No modelo tradicional de perímetro há comunicação entre os dispositivos cliente e servidor antes de efetuarem a autenticação. A *Cloud Security Alliance* (CSA) propôs o Perímetro Definido por *Software* para autenticar antes de a primeira comunicação acontecer, começando com disponibilidade e visibilidade zero. É um modelo de segurança para verificar a identidade de dispositivos e usuários antes de conceder acesso. A autorização por um único pacote (*Single Packet Authorization* - SPA) é o primeiro passo para acesso a um SDP.

O SPA não é um substituto para autenticação, apenas outra camada, que permite que IPs estejam acessíveis e portas TCP/UDP sejam abertas quando necessário.

Através da análise do SPA contido no modelo SDP v1.0, consideraram-se pontos que poderiam ser melhorados no modelo. Nesse sentido, fortaleceu-se a autenticidade e originalidade do SPA. Para incremento de robusteza no processo de autenticação, outros seis novos módulos foram criados e incorporados ao processo de criação do SPA: pontualidade, integridade e confidencialidade de dados, compatibilidade com o protocolo IPv6, derivação da senha do usuário e confidencialidade de tráfego. Complementarmente se definiu uma nova arquitetura para alternar o envio do SPA e da conexão TCP subsequente entre os *Controllers front end.* Desta forma, estabeleceu-se a ocultação da conexão TCP subsequente para autenticação inicial de usuários.

No entanto, observou-se que vulnerabilidades ainda persistem quando a identidade de um dispositivo é vinculado ao seu endereço IP.

Nesse sentido, esse trabalho apresentou uma extensão à nova arquitetura SDP para resolver falhas que ainda persistem no modelo TCP/IP com a possibilidade de exploração do *gap* temporal (tempo em que o controlador de perímetro fica aguardando pela conexão TCP do IP informado via SPA para iniciar a autenticação do usuário). A extensão altera o modelo de segurança de perímetro definido por *software*, de uma solução baseada em TCP/IP para uma solução baseada em dispositivos confiáveis. A adoção de *fingerprint* do dispositivo passa a atuar como fator principal na identificação do Cliente e o endereço IP é colocado em segundo plano.

Em síntese, dado as vulnerabilidades existentes no protocolo SDP v1.0, esse trabalho propôs novas definições ao padrão. Definiu-se um novo modelo de criação de um *Single Packet Authorization*. Criou-se uma nova arquitetura para recebimento e processamento do SPA e estabelecimento da conexão TCP subsequente. Incorporou-se ao processo de autenticação do usuário, o *fingerprinting* de dispositivos. Para elevar o grau de segurança, diferentes fatores e métodos de autenticação foram abordados.

Os resultados experimentais demonstram que as soluções propostas contribuem para o aumento dos níveis de proteção e de resiliência do padrão de referência

SDP ao mitigar um maior número de ataques na arquitetura funcional sem prejudicar o desempenho da solução.

7.1 Trabalhos Futuros

Para dar continuidade aos trabalhos realizados nesta dissertação, propõem-se algumas atividades que poderão ser realizadas futuramente:

- Analisar durante o procedimento de *handshaking* TLS, a viabilidade do *fingerprint* (*hash* de 32 *bytes*) fazer parte das informações que o *Controller front end* precisa para se comunicar com o Cliente usando TLS. A sequência de *bytes* (*fingerprint* do dispositivo) permitirá a ambos calcularem a chave secreta a ser usada para a criptografia de dados de mensagens subsequentes;
- Procurar eliminar a necessidade de instalação de um aplicativo Cliente ao criar uma extensão para navegadores *web* com as mesmas funcionalidades;
- Receber um arquivo PAC (*Proxy Auto-Config*) do canal *out-of-band* com as configurações necessárias e atualizadas para cada acesso. As regras contidas no arquivo PAC encaminham automaticamente o SPA e conexão TCP subsequente aos *Controllers front end;*
- Direcionar através da extensão do navegador todo tráfego ao *Controller front end*, que passa a atuar como um *proxy* ou *gateway*. Os usuários somente poderão se comunicar com outras redes, aplicações ou serviços pelo *Controller front end.* Ele disponibiliza os acessos a cada usuário, conforme os devidos níveis de autorização;
- Implementar uma estratégia de proteção adequada para detectar e proteger com rapidez a infraestrutura de recebimento do SPA contra ataques DOS/DDos;
- *Smartphone* ou *tablet* devem ser capazes de realizarem as mesmas tarefas que as atuais plataformas implementadas (*desktops* e *laptops)* e, portanto, portar todos os recursos para dispositivos móveis é recomendado.

REFERÊNCIAS

BAI, L.; LI, S. VLSI implementation of high-speed SHA-256. In: **2009 IEEE 8th International Conference on ASIC**. [S.l.: s.n.], 2009. p. 131 – 134. ISSN 2162-7541.

BEIKVERDI, A.; TAN, I. K. T. Improved look-ahead re-synchronization window for HMAC-based one-time password. In: **IET International Conference on Wireless Communications and Applications (ICWCA 2012)**. [S.l.: s.n.], 2012. p. 1 – 5.

BILGER, B. et al. **Cloud Security Alliance - SDP - Specification 1.0**. 2014. Disponível em: <https://cloudsecurityalliance.org/download/sdp-specification-v1-0/>. Acesso em: 20/08/2017.

BOOTH, Y. W.; KUMHYR, D. B. **Method and system for tracing missing network devices using hardware fingerprints**. 2007.

BORTEN, K. Two-factor authentication: Patient privacy and marketing Cybersecurity for todays world. **Briefings on HIPAA (BRIEF HIPAA)**, v. 17, n. 1, p. 1 – 5, January 2017. ISSN 1537-0216. Health Services Administration; USA.

CERT.BR. **Cartilha de Segurança para Internet**. 2017/03/16. Disponível em: <https://cartilha.cert.br/ataques/>. Acesso em: 21/08/2018.

CHUAH, C. W.; DAWSON, E.; SIMPSON, L. Key Derivation Function: The SCKDF Scheme. In: **Security and Privacy Protection in Information Processing Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 125 – 138.

COOPER, A. et al. **Privacy Considerations for Internet Protocols**. RFC Editor, 2013. RFC 6973. Disponível em: <https://rfc-editor.org/rfc/rfc6973.txt>.

COUNCIL, P. S. S. **Multi-Factor Authentication**. 2017. Version: 1.0. Disponível em: <https://www.pcisecuritystandards.org/pdfs/Multi-Factor-Authentication-Guidance-v1.pdf>. Acesso em: 15/08/2018.

DAEMEN, J.; RIJMEN, V. **AES Proposal**: Rijndael. 1999.

DESAI, A. et al. Parallelization of AES algorithm for disk encryption using CBC and ICBC modes. In: **2013 Fourth (ICCCNT)**. [S.l.: s.n.], 2013. p. 1 – 7.

DOULIGERIS, C.; SERPANOS, D. N. **Network Security** : Current status and future directions. [S.l.]: IEEE PRESS, 2007. ISSN 9780470099742. ISBN 978-0-471-70355-6.

ETCHEGOYEN, C. S. **Authentication of computing and communications hardware**. [S.l.], 2014.

FOROUZAN, B. A. Data Communications and Networking. In: ____. **McGraw-Hill Companies, Inc.** 4 th. ed. [S.l.]: McGraw-Hill, 2007. cap. 31.6. ISBN 0-07-296775-7.

GAO, K.; CORBETT, C.; BEYAH, R. A passive approach to wireless device fingerprinting. In: **2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN)**. [S.l.: s.n.], 2010. p. 383 – 392. ISSN 1530-0889.

GARDNER, P. B.; VOLODARETS, V. **Method for determining identification of an electronic device**. [S.l.], 2017.

GAUTAM, T.; JAIN, A. Analysis of brute force attack using TG â€" Dataset. In: **2015 SAI Intelligent Systems Conference (IntelliSys)**. [S.l.: s.n.], 2015. p. 984 – 988.

GROSSE, E.; UPADHYAY, M. Authentication at Scale. **IEEE Security Privacy**, v. 11, n. 1, p. 15 – 22, 2013. ISSN 1540-7993.

HAGEN, J. T.; MULLINS, B. E. TCP veto: A novel network attack and its Application to SCADA protocols. In: **2013 IEEE PES Innovative Smart Grid Technologies Conference (ISGT)**. [S.l.: s.n.], 2013. p. 1 – 6.

HANSEN, T.; 3RD, D. E. E. **US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)**. RFC Editor, 2011. RFC 6234. Disponível em: <https://rfc-editor.org/rfc/rfc6234.txt>.

HASAN, R.; KHAN, R. Unified authentication factors and fuzzy service access using interaction provenance. **Computers & Security**, v. 67, p. 211 – 231, 2017. ISSN 0167-4048. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167404817300408>.

Information Technology Laboratory. **Secure Hash Standard (SHS)**. Gaithersburg, 2015. Disponível em: <http://dx.doi.org/10.6028/NIST.FIPS.180-4>.

IRFAN, K. et al. Text based graphical password system to obscure shoulder surfing. In: **2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST)**. [S.l.: s.n.], 2018. p. 422 – 426. ISSN 2151-1411.

JOSEFSSON, S. **The Base16, Base32, and Base64 Data Encodings**. RFC Editor, 2006. RFC 4648. Disponível em: <https://rfc-editor.org/rfc/rfc4648.txt>.

KALISKI, B. **PKCS #5**: Password-based cryptography specification version 2.0. RFC Editor, 2000. RFC 2898. Disponível em: <https://rfc-editor.org/rfc/rfc2898.txt>.

KEARNS, D. Authentication factors; * Highlights to the upcoming European Identity Conference. **Network World [0887-7661]**, Academic OneFile, February 2009. Disponível em: <http://go.galegroup.com/ps/i.do?p=AONE&sw=w&u=capes&v=2.1&id=GALE|A214585389&it=r&asid=de6ebb34fd512d6adbc5e10256528392>. Acesso em: 15 Sept. 2017.

KUMAR, R.; TALWAR, I. Network Security using Firewall and Cryptographic Authentication. **International Journal of Computer Applications (0975 – 8887)**, v. 57, n. 23, November 2012.

LEE, Y. S.; LIM, H.; LEE, H. A study on efficient OTP generation using stream cipher with random digit. In: **2010 The 12th Inter. Conf. on Advanced Communication Technology (ICACT)**. [S.l.: s.n.], 2010. v. 2, p. 1670 – 1675. ISSN 1738-9445.

LI, J. Design of authentication protocols preventing replay attacks. In: **2009 International Conference on Future BioMedical Information Engineering (FBIE)**. [S.l.: s.n.], 2009. p. 362 – 365. ISSN 2157-9598.

LIEW, J. et al. One-Time Knocking framework using SPA and IPsec. In: **2010 2nd International Conference on Education Technology and Computer**. [S.l.: s.n.], 2010. v. 5, p. V5 – 213. ISSN 2155-1812.

LUANGMANEEROTE, S.; ZALUSKA, E.; CARR, L. Inhibiting Browser Fingerprinting and Tracking. In: **2017 IEEE 3rd international conference on big data security on cloud**. [S.l.: s.n.], 2017. p. 63 – 68.

NAKAMURA, E. T.; GEUS, P. L. de. **Segurança de Redes em ambientes cooperativos**. 1. ed. São Paulo: Novatec Editora, 2007. ISBN 978-85-7522-136-5.

NARAYANASWAMY, K. **Dynamic access control policy with port restrictions for a network security appliance**. 2008. Patent US8572717B2. Disponível em: <https://patents.google.com/patent/US8572717B2/en>.

NEWMAN, C.; KLYNE, G. **Date and Time on the Internet**: Timestamps. RFC Editor, 2002. RFC 3339. Disponível em: <https://rfc-editor.org/rfc/rfc3339.txt>.

NOR, F. B. M.; JALIL, K. A.; MANAN, J. A. An enhanced remote authentication scheme to mitigate man-in-the-browser attacks. In: **Proceedings Title**: 2012 international conference on cyber security, cyber warfare and digital forensic (cybersec). [S.l.: s.n.], 2012. p. 271 – 276.

NOURELDIEN, N. A.; HUSSEIN, M. O. Block Spoofed Packets at Source (BSPS): A method for detecting and preventing all types of spoofed source IP packets and SYN flooding packets at source: A theoretical framework. In: **2009 Second International Conference on the Applications of Digital Information and Web Technologies**. [S.l.: s.n.], 2009. p. 579 – 583.

OSBORN, B. et al. BeyondCorp: Design to Deployment at Google. **;login:**, v. 41, p. 28 – 34, 2016. Disponível em: <https://www.usenix.org/publications/login/spring2016/osborn>.

PETERSON, L. L.; DAVIE, B. S. **Redes de Computadores - uma abordagem de sistemas**. 5º. ed. [S.l.]: Elsevier, 2013. ISBN 978-85-352-4897-5.

PUTHAL, D. et al. Building Security Perimeters to Protect Network Systems Against Cyber Threats [Future Directions]. **IEEE Consumer Electronics Magazine**, v. 6, n. 4, p. 24 – 27, 2017. ISSN 2162-2248.

QASIM, Z. A. K. andNadeem Javaid andM. H. Arshad andAyesha Bibi andB. Performance Evaluation of Widely used Portknoking Algorithms. **CoRR**, abs/1207.1700, 2012. Disponível em: <http://arxiv.org/abs/1207.1700>.

RAHMAN, T.; SHUVA, T. F.; ALI, K. M. A. Trusted Device along with Trusted Location and Biometry based Authentication Method. **International Journal of Computer Applications (0975 – 8887)**, v. 150, n. 4, p. 26 – 30, September 2016. Foundation of Computer Science (FCS), NY, USA.

RASH, M. **Single Packet Authorization with Fwknop, A Comprehensive Guide to Strong Service Concealment with fwknop**. 2016. Disponível em: <http://www.cipherdyne.org/fwknop/docs/fwknop-tutorial.html>. Acesso em: 13/11/2017.

ROWLAND, C. et al. **Generating globally unique device identification**. 2008.

SHEN, C. et al. Passive fingerprinting for wireless devices: A multi-level decision approach. In: **2017 IEEE International Conference on Identity, Security and Behavior Analysis (ISBA)**. [S.l.: s.n.], 2017. p. 1 – 6.

SOUZA, D. F. L.; BURLAMAQUI, A. M. F.; SOUZA FILHO, G. L. A Multi Factor Authentication Approach Based on Biometrics, Optical Interference and Chaotic Maps. **IEEE Latin America Transactions**, IEEE, v. 15, p. 1900 – 1908, September 2017. ISSN 1548-0992.

SPEAR, B. et al. Beyond Corp: The Access Proxy. **Login**, v. 41, n. 04, p. 28 – 33, 2016.

STEVENS, M. et al. The first collision for full SHA-1. **CWI Amsterdam and Google Research**, p. 1 – 23, 2017. Disponível em: <http://shattered.io/static/shattered.pdf>.

TAHIR, R. et al. Resilience against brute force and rainbow table attacks using strong ICMetrics session key pairs. In: **2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)**. [S.l.: s.n.], 2013. p. 1 – 6.

TARIQ, M.; BAIG, M. S.; SAEED, M. T. Associating the Authentication and Connection-Establishment Phases in Passive Authorization Techniques. **World Congress on Engineering**, London, U.K., I, July 2008. ISSN 978-988-98671-9-5.

VIEW, M. et al. **HOTP**: An hmac-based one-time password algorithm. RFC Editor, 2005. RFC 4226. Disponível em: <https://rfc-editor.org/rfc/rfc4226.txt>.

VIEW, M. et al. **TOTP**: Time-based one-time password algorithm. RFC Editor, 2011. RFC 6238. Disponível em: <https://rfc-editor.org/rfc/rfc6238.txt>.

VILLELA, A. D. A. **Access control system based on a hardware and software signature of a requesting device**. [S.l.], 2007.

WARD, R.; BEYER, B. BeyondCorp: A New Approach to Enterprise Security. **;login:**, Vol. 39, No. 6, p. 6 – 11, 2014.

ZHAO, J. X. Research and Design on an Improved TOTP Authentication. In: **Information Technology Applications in Industry II**. [S.l.]: Trans Tech Publications, 2013. (Applied Mechanics and Materials, v. 411), p. 595 – 599.

ZHU, L.; YU, S.; ZHANG, X. Improvement upon Mutual Password Authentication Scheme. In: **2008 International Seminar on Business and Information Management**. [S.l.: s.n.], 2008. v. 1, p. 400 – 403.

ZORKTA, H.; ALMUTLAQ, B. Harden Single Packet Authentication (HSPA). **International Journal of Computer Theory and Engineering**, v. 4, n. 5, p. 717 – 721, 2012.

Apêndices

Artigo Completo Publicado em Evento Internacional:

LUCION, E. L. R.; NUNES, R. C. **Software Defined Perimeter: improvements in the security of Single Packet Authorization and user authentication**. In: (CLEI), C. L. A. de Estudos de I. (Ed.). Simpósio latino-americano em Infraestrutura, Hardware e Software (SLIHS). São Paulo: [s.n.], 1º a 5 de Outrubro de 2018. Disponível em: <http://cleilaclo2018.mackenzie.br/docs/SIHS/182649.pdf>.