Incrementando os níveis de segurança na autenticação com Single Packet Authorization e Device Fingerprinting

Everson Luis Rosa Lucion [1], Raul Ceretta Nunes [1]

[1] Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal de Santa Maria (UFSM) - Santa Maria - RS - Brasil

everson@cpd.ufsm.br, ceretta@inf.ufsm.br

Abstract. *A new approach to network perimeter control is to authenticate before the first communication happens, as proposed in the Software Defined Perimeter (SDP). The use of Single Packet Authorization (SPA) in SDP is critical for first access to occur only after device authentication. However, the subsequent TCP connection problem may persist in SPA authentication techniques when authentication is bound to the IP address, such as in the SDP SPA. This work proposes a new model for creating and sending the SPA in the SDP. The new model includes in the SPA framework a device fingerprint field. Also, a method for constructing and using the new fingerprint field is proposed in order to solve the temporal gap between SPA authentication and the subsequent connection for user authentication. The results demonstrate that the proposed solution mitigates improper access and considerably increases the degree of difficulty in detecting, replicating or reading the 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 usability of the solution.*

Resumo. *Uma nova abordagem para controle de perímetro de rede é autenticar antes de a primeira comunicação acontecer, tal como proposto no Software Defined Perimeter (SDP). O uso de Single Packet Authorization (SPA) no SDP é fundamental para que o primeiro acesso ocorra apenas após a autenticação do dispositivo. Entretanto, o problema da conexão TCP subsequente pode persistir em técnicas de autenticação via SPA quando a autenticação está vinculada ao endereço IP, tal como no SPA do SDP. Este trabalho propõe um novo modelo para criação e envio do SPA no SDP. O novo modelo inclui, na estrutura do SPA, um campo de fingerprint de dispositivo. Também é proposto um método para construir e usar o novo campo de fingerprint, a fim de solucionar o gap temporal entre a autenticação do SPA e a conexão subsequente para autenticação do usuário. Os resultados demonstram que a solução proposta combate o acesso indevido 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 usabilidade da solução.*

1. Introdução

O modelo tradicional de perímetro de rede baseado em *firewall*, amplamente utilizado para proteger e bloquear o acesso a recursos internos da infraestrutura de Tecnologia da

Informação (TI), funciona bem quando todos os colaboradores trabalham, exclusivamente, em construções pertencentes a uma mesma empresa [Ward e Beyer, 2014]. Porém, as atuais infraestruturas de TI estão se tornando mais híbridas[1] e diversificadas, havendo uma tendência de migração de infraestrutura baseada em *hardware* para a de *software* [Puthal *et al.*, 2017].

Um modelo de controle de perímetro de rede baseado em *software* foi proposto pela *Cloud Security Alliance*[2] (CSA) [Bilger *et al.*, 2014]. O modelo procura oferecer aos proprietários de aplicações a capacidade de implantar funcionalidades perimetrais para autenticar antes de a primeira comunicação acontecer e, é referido como Perímetro Definido por Software (*Software Defined Perimeter* - SDP). No modelo, a autorização por um único pacote (*Single Packet Authorization* - SPA) é o passo inicial para ter acesso a um SDP.

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. Durante esse tempo, o dispositivo cliente pode conectar-se ao servidor [Zorkta e Almutlaq, 2012].

Embora o SDP tenha oferecido uma solução que permita fortalecer o controle de perímetro de rede, em Lucion e Nunes [2018], a análise da Especificação 1.0 [Bilger *et al.*, 2014] demonstrou que questões de segurança precisam ser melhoradas ou abordadas durante o processo de sua criação e envio do SPA: 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 [M'Raihi *et al.*, 2005] para formação da senha única a torna válida por tempo indeterminado; 4) a coexistência entre IPv4 e IPv6 não é clara; 5) a senha secreta do usuário usada não oferece proteção contra ataques de força bruta ou dicionário; 6) a verificação de integridade de todos os campos do *payload* não foi prevista; 7) a autenticidade de dados deveria ser prevista e aplicada a todo *payload*; e 8) ações que atendam a confidencialidade de tráfego não foram previstas.

Um problema chave, que assola o SDP, é o da conexão TCP subsequente que pode persistir em técnicas de autenticação via SPA. O problema ocorre quando a identidade é vinculada a um endereço IP, pois é assumido que, apenas, o dispositivo verdadeiro acessará um servidor a partir deste endereço IP. Por exemplo, a tradução de endereços de rede (*Network Address Translation* - NAT), *proxy* e técnicas similares têm um mesmo IP compartilhado centenas ou milhares de vezes, bem como ataques de IP *spoofing* também podem ocorrer. Portanto, nesse 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 anteriormente via SPA.

Este trabalho propõe um novo modelo de SPA com *Device Fingerprint* para fortalecer os níveis de segurança no processo de autenticação SDP. O ponto chave da proposição é que a validação do dispositivo deve fazer parte do processo de autenticação para mitigar

[1] 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.

[2] 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/.*

ataques de roubo de credenciais. A contribuição é um novo modelo para criação e envio do SPA no SDP e a proposição de um método para construção e uso de *figerprint* para solucionar o *gap* temporal entre a autenticação do SPA e a conexão subsequente para autenticação do usuário.

O artigo está organizado da seguinte forma: a seção 2 apresenta os trabalhos relacionados e a seção 3 descreve como é realizada a autenticação no SDP. A seção 4 apresenta as melhorias na segurança do *Single Packet Authorization.* Já, a seção 5, detalha o novo método para autenticação com *Device Fingerprint no SPA*. Na seção 6 é apresentado a análise quantitativa e, na seção 7 a validação da segurança e, por fim, a seção 8 traz as conclusões.

2. Trabalhos Relacionados

Os estudos [Tariq *et al.*, 2008; Liew *et al.*, 2010; Zorkta e Almutlaq, 2012; Kumar e 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 *et al.*, 2008; Liew *et al.*, 2010; Zorkta e Almutlaq, 2012; Kumar e Talwar, 2012]. A integridade de dados foi fortemente usada em [Zorkta e Almutlaq, 2012] e parcialmente [Tariq *et al.*, 2008; Liew *et al.*, 2010; Kumar e Talwar, 2012]. A autenticidade foi abordada em [Tariq *et al.*, 2008; Liew *et al.*, 2010; Zorkta e Almutlaq, 2012; Kumar e Talwar, 2012]. A derivação de chaves foi, somente, usada em [Liew *et al.*, 2010]. Nestes trabalhos, não houve considerações sobre a compatibilidade com o protocolo IPv6 e a confidencialidade de tráfego.

Em [Kumar e Talwar, 2012]é explorado *esteganografia* nos campos do *header* TCP SYN para embutir o código *hash* resultante da autenticação, bem como o SPA e a conexão TCP subsequente são tratados em um único pacote enviado. Em [Liew *et al.*, 2010] é proposto um canal *out-of-band*[3] (SMS) para receber dados para envio do SPA e criação da VPN subsequente. Em [Tariq *et al.*, 2008] o servidor somente aceita o TCP SYN caso o SPA contenha um *nonce* embutido. Em [Zorkta e Almutlaq, 2012] o cliente envia dentro do SPA a porta em que irá se conectar via TCP. Enfim, destes trabalhos observa-se que os autores não abordaram todos os aspectos de segurança de forma equânime tal como nesse trabalho, que busca-se prover um 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. Diferente dos demais, este trabalho define também um processo de *fingerprinting* para mitigar o problema da conexão TCP subsequente, sem alterar a estrutura do protocolo TCP/IP.

3. Autenticação no SDP

O TCP permite uma primeira comunicação (*e.g., Three-way Handshake*[4]) entre dispositivos antes de efetuarem a autenticação, o que deixa uma falha de segurança para que um atacante adentre no sistema, antes de comprovar suas credenciais [Puthal *et al.*, 2017]. A ideia chave

[3]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.

[4]Handshake de três vias (SYN, SYN+ACK, ACK). É responsável pelo estabelecimento de conexões no TCP.

do SDP é eliminar a fraqueza do protocolo TCP/IP, no que diz respeito, ao estabelecimento de conexão antes do processo de autenticação. No SDP, o SPA é o passo inicial para ter acesso ao perímetro.

A Figura 1 ilustra como o SPA é elaborado pela Especificação 1.0 [Bilger *et al.*, 2014]. Observa-se que o resultado do algoritmo 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* (controla o acesso ao SDP).

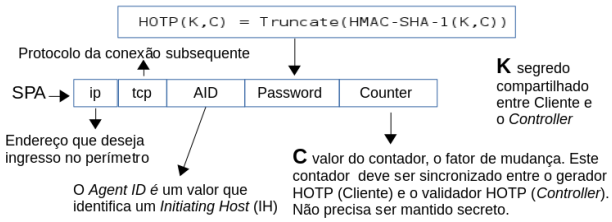

Figura 1. Estrutura original do SPA definido pela Especificação 1.0.

Na arquitetura SDP proposta por [Bilger *et al.*, 2014], 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*).

Como resultado, o modelo, oferece a ocultação da infraestrutura, de serviços e controles de acesso, bem como a confidencialidade e integridade das comunicações.

4. Melhorias na segurança do *Single Packet Authorization*

Essa seção propõe uma nova abordagem projetada sob aspectos modulares de criação do SPA, tendo como base o proposto pela Especificação 1.0 [Bilger *et al.*, 2014]. Para que os dados transmitidos não sejam comprometidos, segundo [Peterson e Davie, 2013] é recomendável abordar os seguintes aspectos de segurança: confidencialidade de dados e de tráfego, integridade e autenticidade, originalidade e pontualidade. Os módulos de originalidade e autenticidade foram fortalecidos, pois, já faziam parte do protocolo. Seis novos módulos foram criados: pontualidade, integridade, confidencialidade de dados, derivação de chaves, compatibilidade com o protocolo IPv6 e confidencialidade de tráfego. Os módulos projetados são detalhados a seguir.

4.1. Módulo 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) [Chuah *et al.*, 2013]. A *Password-Based Key Derivation Function* 2 ou função PBKDF2 - RFC 2898 [Kaliski, 2000] usa o HMAC-SHA-256 - RFC 6234 [Hansen e Eastlake, 2011] como *Pseudorandom Function* (PRF).

A ideia por trás do PBKDF2 e de qualquer algoritmo 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. A função de derivação de chaves usada é:

$$Key^{(x)} = \text{(HMAC-SHA-256 (}password,\ \ salt\text{, 32))}$$

4.2. Módulo Pontualidade

Um protocolo que detecta técnicas de atraso fornece pontualidade. É necessário incluir um leve carimbo de tempo[5] 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. Neste sentido, o campo *timestamp*, conhecido como *Current Unix Timestamp* - RFC 3339 [Newman e 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 intervalo permitido.

4.3. Módulo Originalidade

Um atacante poderia capturar e retransmitir uma cópia de um SPA. Objetivando aumento da segurança contra ataques de repetição, força bruta ou dicionário, este trabalho, uniu os protocolos HOTP - RFC 4226 [M'Raihi *et al.*, 2005] e TOTP - RFC 6238 [M'Raihi *et al.*, 2011] para criar a função **HTOTP**. Têm-se senhas de uso único baseado em eventos (*counter* - mantido simétrico entre o cliente e o Controller) e tempo humano (*timestamp*). Incorporar um *counter* para cada usuário, sua senha simétrica *Key1,* resultante da equação $Key1^{(1000)} = (HmacSha256(passwordUser \parallel salt1 \parallel 32))$ e o valor do *timestamp*, torna ataques de *replay* praticamente ineficazes. A Figura 2 apresenta detalhes da nova função criada.

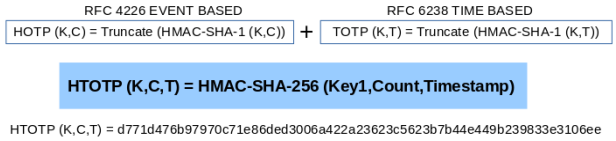

Figura 2. Função HTOTP criada e exemplo de *hash* resultante de 32 *bytes*.

4.4. 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 e Davie, 2013]. Um autenticador é um valor incluído em um SPA a ser 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].

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

A Especificação 1.0 [Bilger *et al.*, 2014] usa o *Secure Hash Algorithm* 1 (SHA-1) no HMAC aplicado somente ao campo *counter* e passwd para prover Autenticidade de Dados. Com o objetivo de melhorar a segurança é proposto o uso do *Secure Hash Algorithm* SHA-256 aplicado a todos os campos do *payload*. A Figura 3 apresenta o modelo implementado, onde o *payload* do SPA criado possui os seguintes campos: *username* (UUID do usuário), HTOTP (*hash* resultante OTP), *Counter* (contador baseado em eventos), *Timestamp* (tempo humano), IPv4/IPv6 (endereço IP que deseja ingressar no perímetro) e *Key1*.

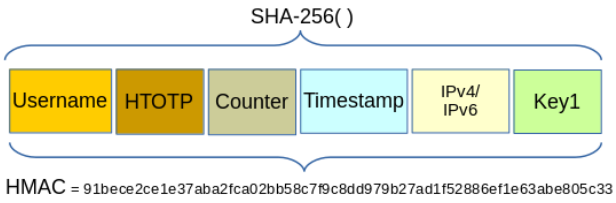

Figura 3. Aplicação de integridade e autenticidade e exemplo de *hash*.

O resultado da função SHA-256, implementada via biblioteca *openssl/sha.h*[6], retorna um HMAC de 32 *bytes*, que será novamente incorporado ao *payload* para ser criptografado com os outros campos.

4.5. Módulo de Confidencialidade de Dados

Na atual abordagem da Especificação 1.0 [Bilger *et al.*, 2014], o SPA é enviado às claras. No entanto, é prático e possível cifrar os dados transmitidos de modo a impedir que um atacante entenda o conteúdo, conforme a Figura 4.

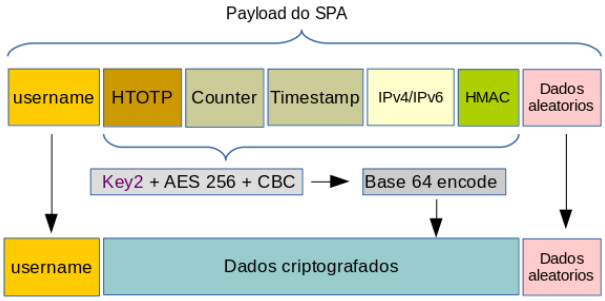

Figura 4. Aplicação da criptografia ao *payload* do SPA.

Neste módulo, o campo HMAC (*hash* resultante do módulo Integridade e Autenticidade de Dados) é incorporadas ao modelo para ser criptografado com os campos HTOPT, Counter, Timestamp e IPv4/IPv6. A chave obtida para criptografia do *payload* é derivada da equação $Key2^{(2000)} = (HmacSha256(passwordUser \parallel salt2 \parallel 32))$. O campo Dados_aleatorios é usado pelo módulo de confidencialidade de tráfego e o *username* identifica o usuário.

[6]https://www.openssl.org

O Padrão Avançado de Cifração (AES - *Advanced Encription Standard*) se tornou um padrão de cifra para chave simétrica. Chamado originalmente de *Rijndael* [Daemen e 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 e Davie, 2013; Desai *et al.*, 2013]. O resultado da criptografia (caracteres não imprimíveis) foram convertidos em caracteres imprimíveis de *Base64*[7] antes de envio ao *Controller* pela aplicação cliente para garantir que não haja corrupção de dados.

4.6. Módulo Compatibilidade IPv4 e IPv6

As conexões TCP/IP podem se originar nas redes IPv4 ou IPv6. A compatibilidade com o protocolo IPv6 foi implementada para que toda e qualquer funcionalidade executada em IPv4 deva ser realizada também em IPv6. Nesse cenário, o *Controller* implementa o conceito de pilha dupla ou *dual stack*[8].

4.7. 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. As seguintes funcionalidades que visam a confidencialidade de tráfego foram adicionadas ao modelo:

- O campo "dados_aleatorios" foi adicionado ao *payload* SPA, formado por caracteres de *Base64* gerados aleatoriamente (e.g., entre 0 e 100 *bytes*). Desta forma, o SPA sempre terá tamanhos variados;
- Uso do protocolo UDP (*User Datagram Protocol*). O SPA é enviado em um pacote único;
- O cliente não efetua nenhuma resolução DNS para enviar o SPA ou estabelecer a conexão TCP subsequente;
- Não permitir que o *Controller* responda a qualquer requisição sem antes receber um SPA válido, dando a ideia de buraco negro[9].

5. Método de Autenticação com Device *Fingerprinting*

Essa seção 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 que o controlador de perímetro fica aguardando em sua porta TCP/443 pela conexão do IP informado via SPA para iniciar a autenticação do usuário). Nesse tempo, ataques (e.g., *ip spoofing*) podem ocorrer entre o SPA recebido 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 de *payload* do 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.

[7]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.

[8]Consiste na convivência do IPv4 e do IPv6 nos mesmos equipamentos, de forma nativa, simultâneamente.

[9]No contexto de redes de computadores, é um lugar onde os pacotes enviados para um determinado destino são todos descartados.

5.1. O Processo de *Fingerprinting* de Dispositivos

Nessa seção, é proposto um método para montar *fingerprint* digital a partir de atributos que apresentem relevância na identificação única do dispositivo similarmente aos trabalhos de [Villela, 2007; Booth e Kumhyr, 2007; Rowland *et al.*, 2008; Etchegoyen, 2014; Osborn *et al.*, 2016; Gardner e Volodarets, 2017].

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 Figura 5 apresenta, através de um mecanismo de abstração (entidades e atributos), a modelagem conceitual de criação do *fingerprint.*

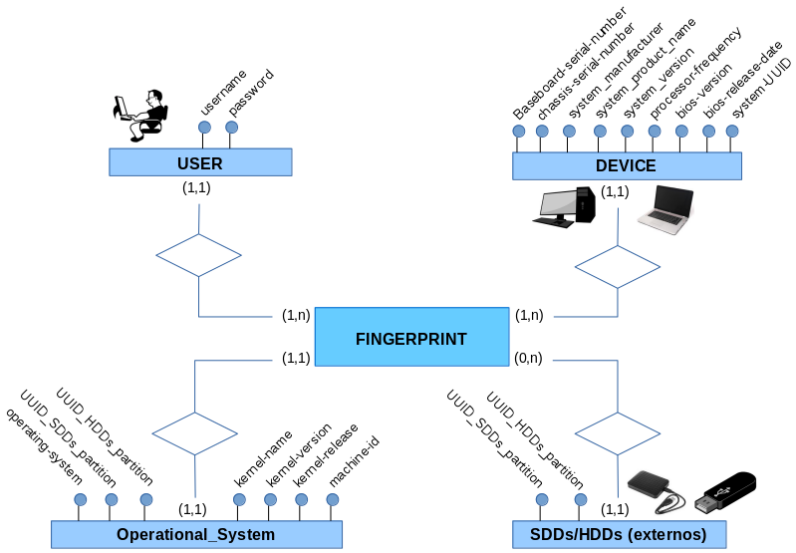

Figura 5. Entidades envolvidas na criação do *fingerprint*.

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

- *User*: dois atributos (*username* e *password*) pertencentes ao usuário;
- *Device*: é uma coleção de componentes físicos ou virtuais com nove atributos;
- *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*: usa o algoritmo de *hash SHA-256*[10] para gerar o *hash* resultante, formando assim, o *fingerprint* do dispositivo.

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

[10] *Secure Hash algorithm* (algoritmo de *hash* seguro) com *hash* resultante de 32 *bytes.*

software [Villela, 2007], um processo [Rowland *et al.*, 2008], serviço de inventário [Spear *et al.*, 2016] ou aplicativo de coleta de atributos [Gardner e Volodarets, 2017]. Todos possuem funcionalidades semelhantes, são instalados no dispositivo cliente, coletam informações e geram um identificador exclusivo.

Este trabalho criou um aplicativo para realizar o processo de *fingerprinting* de dispositivos, além da criação e envio do SPA. O usuário que deseja ingressar no perímetro deve fazer o *download* do aplicativo através de um canal *out-of-band.* Seu uso é aplicado a *desktop* ou *notebook*. Para detalhamento do processo de *fingerprinting*, a Figura 6 mostra o resultado da construção da assinatura digital de um dispositivo computacional e, como ele é adicionado na estrutura de *payload* do SPA.

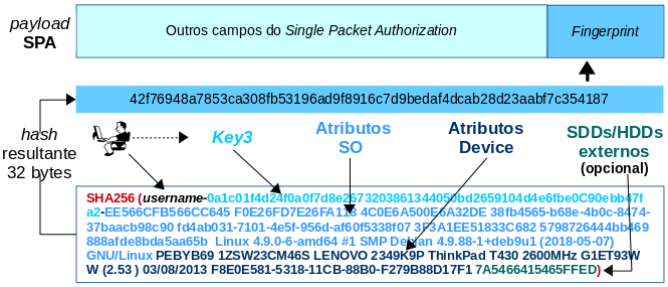

Figura 6. Processo do *fingerprinting* e integração do *hash* com o SPA.

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. Para fortalecer o processo de *fingerprinting*, a senha do usuário (atributo *passorwd*) é usada novamente para derivação de chaves, obtida através da equação $Key3^{(1500)} = (HmacSha256(passwordUser \parallel salt3 \parallel 32))$. A Figura 7 identifica o campo *fingerprint* adicionado ao *payload* SPA para ser criptografado e enviado ao *Controller*.

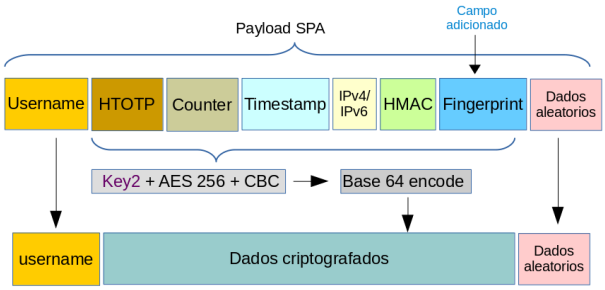

Figura 7. Campo *fingerprint* adicionado ao payload SPA.

O campo *fingerprint* é formado pelo *hash* resultante de informações que caracterizam um dispositivo de forma única. Com isso, o módulos Integridade e Autenticidade de Dados e Confidencialidade de Dados são alterados para suportar o novo campo *fingerprint.* O formato final do SPA a ser enviado ao *Controller* apresenta três campos: *username*, dados_criptografados e dados_aleatórios. Eles são encapsuladas em único pacote, utilizando-se a biblioteca Binn[11].

[11]Disponível em: https://github.com/liteserver/binn

Após o envio do SPA, o aplicativo cliente automaticamente inicia o navegador *web* e realiza uma conexão HTTPS com o *Controller*. Nesse instante é enviado, novamente, o *fingerprint* do dispositivo em sua requisição para validação. A Figura 8 mostra três possíveis respostas ao usuário da requisição de autenticação.

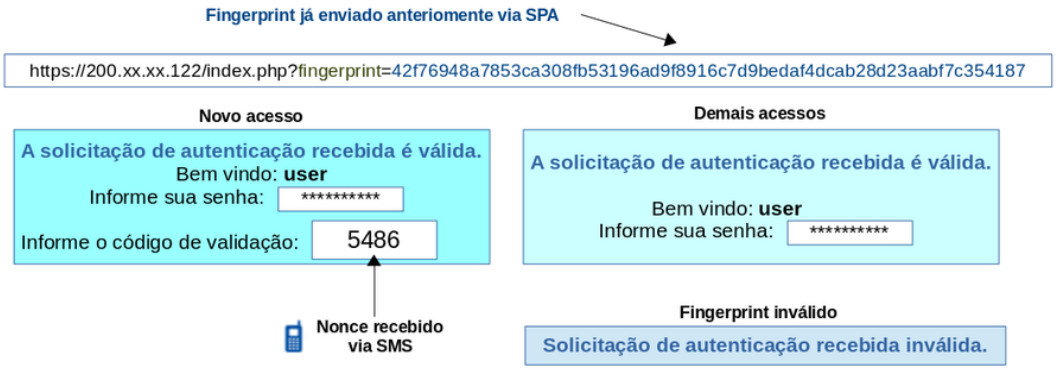

Figura 8. Novo acesso e demais acessos.

Em um novo acesso, o risco é alto e o usuário deve digitar o *nonce* SMS recebido pelo canal *out-of-band*. Nos demais acessos o risco é moderado e apenas a senha é necessária. No caso de *fingerprint* inválido, a solicitação é invalidada. A Figura 9 apresenta o fluxograma do processo de autenticação de usuários via conexão *https*.

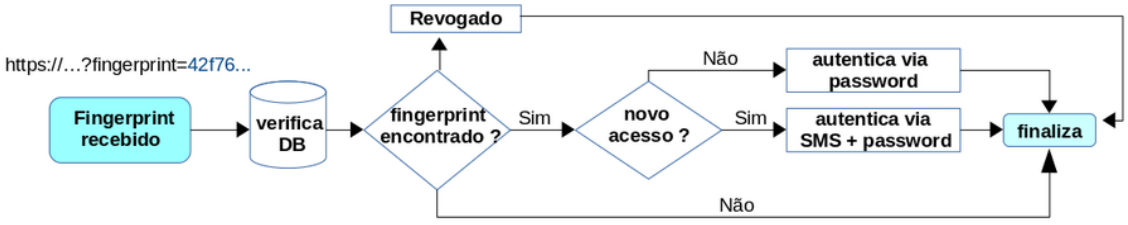

Figura 9. Processo de validação do *fingerprint* de um dispositivo.

Ao receber em uma requisição *https* com *fingerprint* embutido (também enviado anteriormente via SPA) em sua *url*, o *Controller* 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 ou a cada alteração de atributos das entidades) é considerada. É necessário solicitar um fator de autenticação adicional (*nonce* enviado via SMS ao aparelho celular do usuário) através de um canal *out-of-band;*
- validado: *fingerprint*, já utilizado, em acesso anterior. A abordagem de risco moderado é considerada;
- revogado: o dispositivo não tem permissão de acesso ao perímetro, podendo caracterizar um ataque.

Com a autenticação *multi-factor*, adicionam-se camadas (extras) de proteção ao nome de usuário e senha (o que ele sabe). O *nonce* SMS (o que ele tem), o valor da variável *Counter* e *fingerprint* (eventos passados de interação com o sistema) já validado e salvo na base de dados, fornecem maior segurança para ingresso em um SDP.

6. Análise Quantitativa

Nos experimentos quantitativos foram medidos os tempos de processamento em milissegundos à execução de aplicações[12] de testes. A Tabela 1 mostra os resultados médios obtidos durante a criação do SPA e obtenção dos atributos e cálculo do *fingerprint*.

Tabela 1. Resultados da simulação quantitativa em milissegundos.

Fatores de Processamento	Tempo (milissegundos)
Criação do SPA	10,43
Fingerprinting do dispositivo	10,23

Observa-se 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.

7. 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 e Davie, 2013].

Uma vulnerabilidade é definida como uma condição que, explorada por um atacante, pode resultar em uma violação de segurança [CERT.BR, 2017]. 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].

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 e Talwar, 2012; Peterson e Davie, 2013; Zorkta e Almutlaq, 2012; Tariq *et al.*, 2008]. A principal característica dos ataques a serem selecionados foi a possibilidade de que ocorram em uma arquitetura cliente e/ou servidor. A Tabela 2 apresenta os vinte ataques correlacionados com as seguintes colunas:

- **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.
- **Novo SDP**: (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*.

[12]Foram desenvolvidas na linguagem de programação ANSI C.

Tabela 2. Comparativo entre os modelos de defesa de perímetro.

Ataques	Não SDP	SDP v1.0	Novo SDP
Replay attack	vulnerável	2	2
Dictionary e brute-force	vulnerável	vulnerável	4 e 9
Packet crafting	vulnerável	vulnerável	2, 4 e 7
Source IP spoofing	vulnerável	vulnerável	9 e 10
Connection TCP hijacking	vulnerável	vulnerável	9 e 10
Stolen-verifier attack	vulnerável	3	3
Spoofed packets	vulnerável	vulnerável	9 e 10
Man-in-the-middle attack	vulnerável	1,2 e 3	1 ao 10
Port scanning	vulnerável	3	3
DNS spoof	vulnerável	vulnerável	8
DNS cache poisoning	vulnerável	vulnerável	8
DoS e DDoS attack	vulnerável	3	3
Man-in-the-browser	vulnerável	vulnerável	9 e 10
0-day exploit attack	vulnerável	3	3, 9 e 10
Phishing attack	vulnerável	vulnerável	9 e 10
Evesdropping attack	vulnerável	vulnerável	4 e 9
Pharming attack	vulnerável	vulnerável	8, 9 e10
Defacement attack	vulnerável	3	3
Rainbow attack	vulnerável	vulnerável	4
Impersonationattack	vulnerável	vulnerável	9 e 10

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.

Já, o modelo de perímetro descrito pela Especificação 1.0 do SDP apresenta um desempenho de segurança satisfatório. 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 e resolve o problema da conexão TCP/IP subsequente.

8. Conclusões

No modelo tradicional de perímetro há comunicação entre o dispositivo cliente e servidor antes de efetuarem a autenticação, o que geram vulnerabilidades exploradas por diferentes categorias de ataques. A CSA propôs o SDP, um modelo para autenticar antes de a primeira comunicação acontecer, ou seja, um modelo de segurança (com disponibilidade e visibilidade zero) para verificar a identidade de dispositivos e usuários, antes de conceder acesso ao servidor ou serviço. No modelo, 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, mas apenas outra camada, que permite que IPs estejam acessíveis e portas TCP/UDP sejam abertas quando necessário.

Esse trabalho destacou que o SPA contido no modelo SDP v1.0 apresenta vulnerabilidades. Assim, apresentou mecanismos que podem fortalecer a autenticidade e originalidade, já presentes no modelo. Numa abordagem modular, o trabalho também, apresentou soluções para incremento de robusteza no processo de autenticação via SPA. Seis novos módulos foram incorporados ao processo de criação do SPA: pontualidade, integridade e confidencialidade de dados, compatibilidade entre IPv4 e IPv6, derivação da senha do usuário e confidencialidade de tráfego. No entanto, vulnerabilidades ainda persistem quando a identidade de um dispositivo é vinculado ao seu endereço IP. Neste sentido, esse trabalho propôs e demonstrou que a inclusão de um campo de *device fingerprint* ao SPA é uma solução de baixo custo e resolve o problema.

A solução proposta se mostrou eficaz na solução de vulnerabilidades que, ainda, persistiam no SDP 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). Basicamente, a solução 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. O endereço IP é colocado em segundo plano, resolvendo o problema da conexão TCP subsequente. Em síntese, esse trabalho apresentou uma solução que torna mais seguro o padrão SDP, através de um novo modelo de criação e uso do SPA.

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.

Referências

Bilger, B., Boehme, A., Flores, B., e Guterman, Z. (2014). Cloud Security Alliance - SDP - Specification 1.0.

Booth, Y. W. e Kumhyr, D. B. (2007). Method and system for tracing missing network devices using hardware fingerprints. In *Patent No.: US 7,181,195 B2*. USA Patent.

CERT.BR (2017). Cartilha de Segurança para Internet.

Chuah, C. W., Dawson, E., e Simpson, L. (2013). Key Derivation Function: The SCKDF Scheme. In *Security and Privacy Protection in Information Processing Systems*, pages 125 – 138, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cooper, A., Tschofenig, H., Ph.D., D. B. D. A., Peterson, J., Morris, J. B., Hansen, M., e Smith, R. (2013). Privacy Considerations for Internet Protocols. RFC 6973.

Daemen, J. e Rijmen, V. (1999). AES Proposal: Rijndael. In *The Rijndael Block Cipher - Document Version 2 - AES Proposal*, pages 1 – 45.

Desai, A., Ankalgi, K., Yamanur, H., e Navalgund, S. S. (2013). Parallelization of AES algorithm for disk encryption using CBC and ICBC modes. In *2013 Fourth (ICCCNT)*, pages 1 – 7.

Etchegoyen, C. S. (2014). Authentication of computing and communications hardware. In *Patent US 8726407 B2*. United States Patent.

Gardner, P. B. e Volodarets, V. (2017). Method for determining identification of an electronic device. In *Patent No.: US 9,547,780 B2*. United States Patent.

Hansen, T. e Eastlake, D. (2011). *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)*. Internet Engineering Task Force (IETF).

Kaliski, B. (2000). *PKCS #5: Password-Based Cryptography Specification Version 2.0*. Internet Engineering Task Force (IETF).

Kumar, R. e Talwar, I. (2012). Network Security using Firewall and Cryptographic Authentication. *International Journal of Computer Applications (0975 – 8887)*, **57**(23).

Liew, J., Lee, S., Ong, I., Lee, H., e Lim, H. (2010). One-Time Knocking framework using SPA and IPsec. In *2010 2nd International Conference on Education Technology and Computer*, volume 5, pages V5 – 213.

Lucion, E. L. R. e Nunes, R. C. (2018). Software Defined Perimeter: improvements in the security of Single Packet Authorization and user authentication. In *XLIV Conferência Latino-americana de Informática - CLEI*, São Paulo.

M'Raihi, D., Hoornaert, F., Naccache, D., Bellare, M., e Ranen, O. (2005). *HOTP: An HMAC-Based One-Time Password Algorithm*. Internet Engineering Task Force (IETF).

M'Raihi, D., Machani, S., Pei, M., e Rydell, J. (2011). *TOTP: Time-Based One-Time Password Algorithm*. Internet Engineering Task Force (IETF).

Newman, C. e Klyne, G. (2002). Date and Time on the Internet: Timestamps. RFC 3339.

Osborn, B., McWilliams, J., Beyer, B., e Saltonstall, M. (2016). BeyondCorp: Design to Deployment at Google. *;login:*, **41**, 28 – 34.

Peterson, L. L. e Davie, B. S. (2013). *Redes de Computadores - uma abordagem de sistemas*. Elsevier, 5º edition.

Puthal, D., Mohanty, S. P., Nanda, P., e Choppali, U. (2017). Building Security Perimeters to Protect Network Systems Against Cyber Threats [Future Directions]. *IEEE Consumer Electronics Magazine*, **6**(4), 24 – 27.

Rowland, C., Sandford, A., Balakrishnan, S., e McCasey, M. (2008). Generating globally unique device identification. In *Patent No.: US 7428,587 B2*. United States Patent.

Spear, B., Beyer, B. A. E., Cittadini, L., e Saltonstall, M. (2016). Beyond Corp: The Access Proxy. *Login*, **41**(04), 28 – 33.

Tariq, M., Baig, M. S., e Saeed, M. T. (2008). Associating the Authentication and Connection-Establishment Phases in Passive Authorization Techniques. In *Proceedings of the World Congress on Engineering - WCE 2008*, volume I, London, U.K.

Villela, A. D. A. (2007). Access control system based on a hardware and software signature of a requesting device. In *Pub. No.: US 2007/0113090 A1*. United States Patent.

Ward, R. e Beyer, B. (2014). BeyondCorp: A New Approach to Enterprise Security. *;login:*, **Vol. 39, No. 6**, 6 – 11.

Zorkta, H. e Almutlaq, B. (2012). Harden Single Packet Authentication (HSPA). *International Journal of Computer Theory and Engineering*, **4**(5), 717 – 721.