Segurança em Streams Multimídia

Fabrício S. Cardoso[1], Marco Antônio de O. Araújo[1]

[1]Instituto CEUB de Pesquisa e Desenvolvimento (ICPD) – Centro Universitário de Brasília (UniCEUB)
CEP 70.790-075 – Brasília – DF – Brazil

aceyarados@gmail.com, marco.araujo@uniceub.br

Abstract. *This paper proposes a study of security information applied to multimedia streaming. The pursued method of development was such that could assure proper authentication and safe transmission throughout the internet. The history, development and formation of streaming were researched, and various security information concepts were used. For the next step, a search of open source tools that could build the solution took place. The result was a set of tools that work for mutual authentication between client and server, which enabled safe information transmission.*

Resumo. *Este artigo propõe o estudo da segurança da informação aplicada aos fluxos multimídia, ou streams. Buscou-se desenvolver um método que possa assegurar a devida autenticação e transmissão segura do fluxo pela internet. Foram pesquisados a história, desenvolvimento e formação dos fluxos, bem como vários conceitos de segurança da informação. A seguir, partiu-se para a busca de ferramentas que atendessem a uma solução open source. O resultado foi um conjunto de ferramentas que trabalham em função da autenticação mútua entre cliente e servidor, o que habilitou a transmissão segura das informações.*

1. Introdução

No início da era de popularização da internet, as informações disponibilizadas eram, na maior parte, textos e imagens organizados. Em pouco tempo, sua evolução se deu, entre outros tipos, por meio da inserção de conteúdos multimídia em suas páginas. Dessa forma, surgiram os primeiros fluxos – streams – digitais de áudio e vídeo, com o propósito de se publicar conteúdo digital de forma dinâmica e próxima do que o rádio e televisão oferecem. A ideia inicial seria prover um modo de acessar o conteúdo radiofônico ou televisivo sem o uso de aparelhos físicos.

Entretanto, há casos em que esses fluxos digitais podem ser desejáveis apenas para acesso restrito. É o caso de uma rádio ou canal de vídeo digital que possui conteúdo exclusivo e cobra pelo serviço diferenciado que oferece. Com isso, o presente estudo tem como objetivos compreender como se dá a segurança em streams e propor uma solução para o problema.

O presente trabalho está organizado em seis partes. Na primeira e segunda seção, é apresentado o conceito e estrutura de streams multimídia. A terceira seção proporciona uma análise sobre protocolo de streaming utilizado, o Icecast. Na quarta seção, é apresentado como estudo de caso uma solução de segurança implementada com as ferramentas selecionadas. Por fim, as duas últimas partes explicitam os testes realizados e seus resultados.

2. Streams de Áudio

No início dos anos 20, George Squier desenvolveu a primeira rede de transmissão e distribuição de sinais de áudio a partir das linhas de energia elétrica. Isso caracterizou o primeiro registro de uma transmissão de streaming e foi batizada de muzak [1].

Com o passar dos anos, várias tentativas de se transmitir sinais de áudio em fluxo foram feitas. No início da década de 90, a tecnologia de redes de comunicação Multicast Backbone foi utilizada transmitir um show da banda Rolling Stones, e foi considerado o primeiro uso significativo da tecnologia de streaming realizado sobre redes IP [2].

Essas tentativas encontraram um de seus picos em 1995, com a criação da RealAudio [3]. A primeira transmissão foi um jogo de beisebol entre os times New York Yankees e Seattle Mariners. Em 1997, ela criou a primeira solução de streaming de vídeo [7].

O W3C trabalha na revisão da linguagem HTML5. Essa habilita a transmissão de streams de áudio e vídeo apenas com o uso do navegador. A Apple a utiliza como solução padrão para a reprodução de conteúdo de fluxo em seus produtos mais recentes, o que impulsionou o uso da solução ao redor do mundo [8].

3. Estrutura de Streaming

Há quatro elementos que compõem a estrutura de funcionamento de um serviço de streaming. São eles: Captura e Codificação, Serviço, Distribuição e Entrega e Tocador de Mídia [4].

A etapa de captura e codificação consiste em obter o áudio ou vídeo a partir de alguma entrada, gravá-lo em um arquivo de computador, comprimi-lo e codificá-lo. A captura é feita por meio de placas de captura, instaladas em computadores que possuem diversas entradas de áudio e vídeo [4].

Em seguida, comprime-se o arquivo por meio do uso de compressores-decompressores, ou codecs. Estes reduzem o tamanho do arquivo para que não haja perda considerável de qualidade. Além disso, esse passo adequa o arquivo ao tamanho da banda disponível para transmissão [4].

Ao final, insere-se uma camada de identificação ao fluxo. São gravados metadados que carregam informações como duração, número de partes e qualidade. Essa camada é chamada de codificação, ou wrapping [4].

Na figura 1 é possível ver os passos descritos na Captura e Codificação:

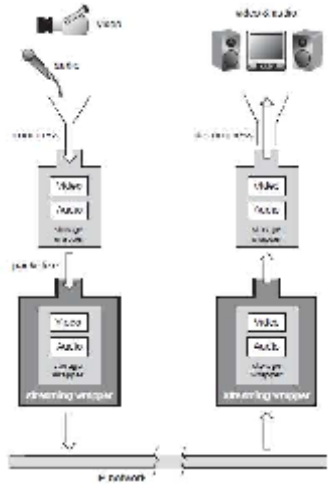

Figura 1. Captura e codificação [4].

Na parte de Serviço, o arquivo gerado é armazenado em um servidor de conteúdo. Ele é responsável pela distribuição do conteúdo pela rede IP. Essa distribuição é regulada, a fim de que o fluxo esteja alinhado às restrições da banda disponível. Além disso, essa parte controla os pedidos de alteração na reprodução, ou seja, fast forwarding – FF – e rewinding – RWD, bem como pausas [4].

Na etapa de Distribuição e Entrega, o foco é alcançar o usuário por meio de um canal de entrega e distribuição. Em regra, é necessária somente a conexão entre o servidor de conteúdo e o tocador de mídia. Entretanto, a internet não foi desenvolvida para esse tipo de transmissão. Logo, perdas de qualidade e pausas inesperadas podem acontecer. Como solução, são necessários algoritmos específicos para correção de erros e regulagem do fluxo, tarefa dessa etapa. Além disso, o aumento na banda disponível contribuiu para melhorar a qualidade da transmissão e potencializar a entrega e distribuição [4].

A última parte é a interface do usuário. O Tocador de Mídia – player – é um aplicativo feito para reprodução de áudio e vídeo em forma de stream ou reprodução de arquivos multimídia. Algumas empresas que disponibilizam esses produtos são a Microsoft, com o Windows Media Player, e a Nullsoft, com o Winamp.

4. O protocolo Icecast

O protocolo utilizado neste trabalho para a transmissão dos fluxos é o Icecast, que é baseado no protocolo shoutcast, concebido pela Nullsoft. Seu funcionamento está baseado no http.

A arquitetura de funcionamento é dividida em três partes [5]:

- Fonte – uma aplicação que gerencia um dispositivo de entrada, como uma câmera ou lista de reprodução de arquivos de mídia;
- Servidor – elemento responsável por receber o que a fonte produz e convertê-lo em fluxo para o cliente;
- Cliente – usado para escutar ou visualizar o conteúdo multimídia a partir do que servidor envia.

Na figura 2, é possível ver o esquema de funcionamento do protocolo Icecast:

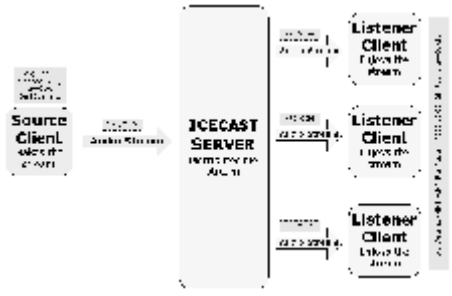

Figura 2. Funcionamento do Icecast [6].

Para que o servidor permita requisições do cliente, ele precisa da fonte. Quando essa conexão é estabelecida, o servidor realiza o trabalho de passar os dados em fluxo da fonte para o cliente. A sequência de passos entre o servidor e o cliente pode ser vista a seguir [5]:

- A fonte solicita uma conexão com a porta de serviço do servidor;
- A fonte envia a senha de conexão definida pelo servidor. Se estiver correta, o servidor responderá com uma mensagem de autorização da conexão e estará pronto para receber os dados. Caso contrário, o servidor sinalizará com uma mensagem de senha inválida;
- Se a fonte receber o acesso, deverá, então, começar a enviar informações sobre o stream para o servidor.

A partir do estabelecimento da conexão mencionada, o fluxo começa a ser transmitido.

5. Proposta de Segurança

A estrutura da implementação está embasada no proxy reverso, representado pela ferramenta Nginx, que tratará a requisição SSL. Ao recebê-la, exigirá o certificado do cliente, funcionalidade opcional do SSL, e o autenticará junto ao certificado da AC para a validação de informações. Da mesma forma, a operação padrão do SSL de autenticação do servidor junto ao cliente será realizada.

Caso haja sucesso, a segurança será reforçada pelo uso da autenticação simples. A solicitação resultante será encaminhada a uma página PHP, que realizará a autenticação mediante o uso de credenciais de usuário e senha. Os dados inseridos serão conferidos por meio de um cadastro armazenado no banco de dados MySQL. Após o sucesso da etapa anterior, uma nova página disponibilizará o acesso a um link que direcionará o usuário para o servidor de mídia, representado pelo Icecast, responsável pela geração e transmissão do fluxo de streaming para o cliente.

O Nginx realizará toda a negociação – handshake – SSL com o cliente, o que evitará que a requisição do streaming chegue diretamente ao servidor de mídia. Ao mesmo tempo, assegurará, com certificados digitais e autenticação simples, que a conexão esteja devidamente autenticada e verificada antes de chegar a seu destino final. Além disso, o servidor é o responsável por hospedar as páginas PHP da autenticação simples.

A configuração do Nginx consiste na edição dos arquivos nginx.conf e default. O primeiro é responsável pelas configurações globais do servidor. O último permite realizar a configuração do host virtual que responderá pelo protocolo https.

O Icecast possui um arquivo de configuração chamado icecast.xml. Nele, definem-se informações como o número de conexões ao servidor de mídia e credenciais de acesso. A fonte de mídia, representada pelo Ices, possui o arquivo de configuração ices.conf. Nele, é possível configurar os dados a respeito do stream gerado. Neste trabalho, foi utilizado um fluxo gerado a partir de uma playlist que será reproduzida dinamicamente.

A geração dos certificados digitais do cliente foi feita com o OpenCA. Esta ferramenta habilita todo o processo de gerenciamento de certificados realizado por uma AC. O arquivo de configuração é o config.xml. Ele define as credenciais de acesso, porta de conexão, entre outros. Para efeito de praticidade, o certificado do servidor SSL foi auto-assinado.

A autenticação simples foi realizada com o uso da linguagem PHP e o banco de dados MySQL. A senha é armazenada no banco de dados na forma de hash SHA-1. A autenticação consiste em solicitar ao usuário, após a validação de seu certificado, usuário e senha. Com isso, caso haja roubo do certificado, o invasor deve ter, ainda, essas credenciais para completar o acesso ao stream.

6. Testes

O ambiente onde foram feitos os testes é composto por cinco máquinas lógicas. A primeira, que hospeda o Nginx, é o proxy reverso e efetua a conexão SSL com autenticação do servidor e do cliente.

A segunda representa os servidores Web e banco de dados que faz a autenticação simples. Esse serviço é também suportado pelo Nginx, que hospeda os scripts PHP.

A terceira máquina é o servidor Icecast, que carrega a lista de reprodução de arquivos mp3 e os transforma em fluxo para entrega ao cliente. As três máquinas estão baseadas na plataforma Ubuntu Linux, instalado no ambiente de virtualização Oracle VirtualBox.

A quarta hospeda o OpenCA, autoridade certificadora que é utilizada para gerar os certificados e roda no sistema operacional CentOS 5.9, também instalado no VirtualBox. A quinta, por sua vez, é a máquina do cliente que deseja acessar o stream por meio do browser.

A figura 3 apresenta um esquema com a solução empregada:

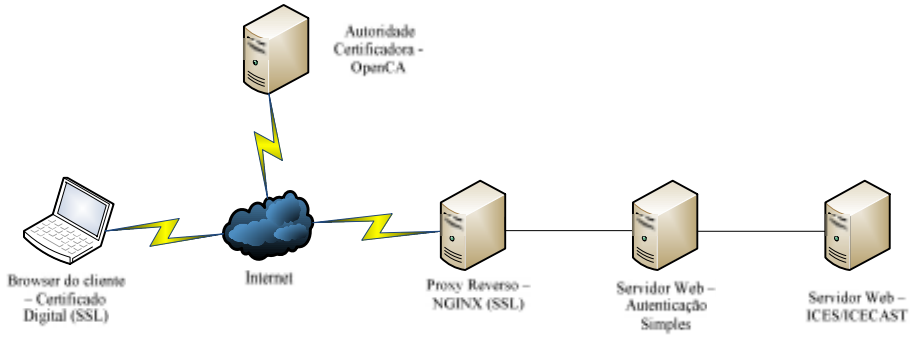

Figura 3. Esquema de funcionamento da solução de segurança.

O primeiro teste consistiu em verificar o funcionamento do processo de autenticação SSL para que o servidor ateste o certificado do cliente. Para isso, foram gerados dois certificados fictícios de cliente, assinados por uma CA fictícia.

Com o acesso https inicial realizado, foi exibida a tela de seleção de certificados, que pode ser vista na figura 4:

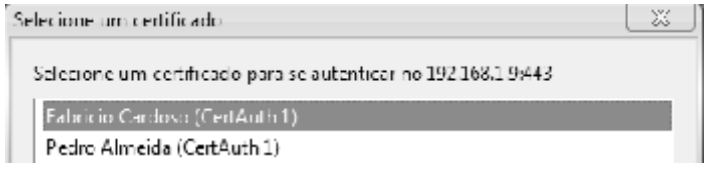

Figura 4. Seleção de Certificado.

O teste inicial foi não apresentar nenhum certificado, clicando no botão "Cancelar" da figura 4. A tela exibida após a ação foi a da figura 5, que bloqueou o acesso caso não fosse apresentado algum certificado:

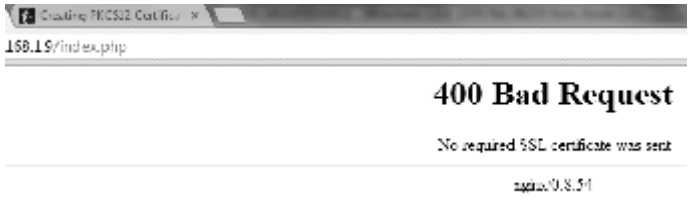

Figura 5. Tela de bloqueio de acesso do Nginx.

Na segunda parte do teste, foi selecionado um certificado. Com isso, o servidor iniciou os procedimentos do handshake SSL para autenticação do cliente.

Utilizou-se o Wireshark para capturar os pacotes de rede e acompanhar o handshake, conforme a tabela 1:

Tabela 1. Extrato do Wireshark mostrando o handshake SSL com autenticação de cliente e servidor.

No.	Source	Destination	Length	Info
23	192.168.1.2	192.168.1.9	220	Client Hello
26	192.168.1.9	192.168.1.2	1514	Server Hello
27	192.168.1.9	192.168.1.2	1514	Certificate
29	192.168.1.9	192.168.1.2	398	Server Key Exchange, Certificate Request, Server Hello Done
52	192.168.1.2	192.168.1.9	1241	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
56	192.168.1.9	192.168.1.2	1004	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
74	192.168.1.2	192.168.1.9	1128	Client Hello
77	192.168.1.9	192.168.1.2	199	Server Hello, Change Cipher Spec, Encrypted Handshake Message
78	192.168.1.2	192.168.1.9	113	Change Cipher Spec, Encrypted Handshake Message
79	192.168.1.2	192.168.1.9	512	Application Data, Application Data
81	192.168.1.9	192.168.1.2	571	Application Data

O primeiro sinal de handshake foi o "Client Hello". O servidor apresentou seu certificado para que o cliente o autenticasse. Em seguida, o servidor exigiu a apresentação de um certificado do cliente. Após a autenticação do servidor pelo cliente, o cliente enviou para o servidor o certificado selecionado. Com isso, uma nova sessão foi criada pelo servidor, mediante a bem sucedida análise de certificado do cliente.

Tentou-se efetuar testes de autenticação do cliente fazendo uso de um certificado emitido pela AC não reconhecida pelo Nginx. O resultado foi o de que o servidor

ignorou certificados que não tinham ACs reconhecidas. Isso pode ser visto na figura 6, em que foi retirado o certificado da AC fictícia das configurações do servidor e houve a substituição por outro, da AC fictícia "serv.dummy.com", e importou-se um certificado emitido por ela, com nome "serv2.dummy.com":

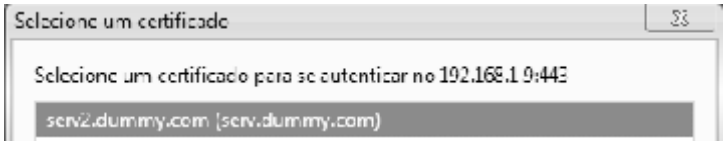

Figura 6. CA substituída e certificado correspondente.

Os certificados emitidos pela CA fictícia não apareceram na lista, o que caracteriza uma boa construção do Nginx quanto à verificação de certificados.

O segundo teste foi elaborado para verificar como funciona a autenticação do servidor pelo cliente. Utilizou-se, em princípio, um certificado auto-assinado pelo servidor Web que hospeda o Nginx. O Google Chrome, ao se deparar com um certificado auto-assinado, emitiu uma tela de aviso alertando para o fato de o certificado do servidor Web não ter sido devidamente verificado. Ao clicar na opção "Continuar mesmo assim", a autenticação foi feita com sucesso. Essa ação não é recomendada e foi tomada apenas para observar o tratamento de certificados auto-assinados.

O último teste foi o de autenticação simples. Se bem sucedido, a aplicação permitirá o acesso ao link do stream. Caso contrário, será negado. Logo após a autenticação de cliente, houve direcionamento para a página PHP, onde foram solicitados usuário e senha. Bem sucedida, houve acesso à página com o link para o fluxo, conforme a figura 7:

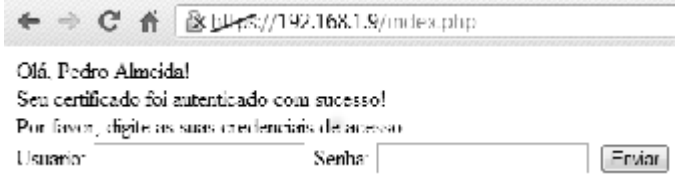

Figura 7. Link de acesso ao stream.

Foi testado, ainda, um perfil que não existe. A tela da figura 8 foi mostrada:

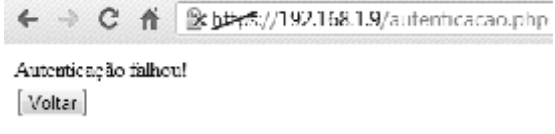

Figura 8. Tela de falha na autenticação simples.

7. Resultados

Conseguiu-se, a partir dos testes, uma observação mais precisa a respeito do comportamento da solução. Na tabela 2, é possível observar as principais características dos testes efetuados:

Tabela 2. Resultados dos testes efetuados.

	Implementação	Aplicabilidade	Personalização	Proteção
Autenticação do cliente	Média	Alta	Sim	Alta
Autenticação do servidor	Fácil	Alta	Sim	Alta
Autenticação simples	Fácil	Alta	Sim	Baixa

O uso de certificados em ambas as partes apresenta um método confiável de autenticação e transmissão de fluxos, pois, de acordo com observações feitas com o Wireshark, o tráfego do stream será criptografado do início ao fim da transmissão. A autenticação simples complementa a autenticação por certificados ao adicionar mais uma camada de segurança. Como extensão, pode-se utilizar algoritmos de hash alternativos, como o SHA-2, para a autenticação simples.

8. Conclusão

Há várias formas de se aplicar a segurança da informação para streams. Neste trabalho, a solução está no canal de comunicação, que é tornado seguro de forma transparente para o usuário. Para que haja acesso ao stream, são necessários certificado e credenciais válidos.

Pode-se ampliar o estudo para várias vertentes, como filtrar tipos de acesso com perfis de assinante, ao usar credenciais de usuário e senha. Além disso, o script em PHP pode utilizar o salt quando a senha for cadastrada. Isso pode lhe conferir maior aleatoriedade, o que eleva a segurança.

Como solução alternativa, há o uso de algoritmos criptográficos diretamente no stream, o que dispensa o SSL. O trabalho pode ser feito bloco a bloco ou em fluxo, com criptografia simétrica ou assimétrica, ou ambas.

Por fim, verificou-se que a solução possui caráter genérico e pode ser utilizada para diversas aplicações, tais como a autenticação de clientes em web services, páginas de acesso restrito, entre outras.

Referências

[1]LANZA, J. Elevator Music. Michigan: The University of Michigan Press, 2004, p. 25-27.

[2]MBONE. Low-complexity Video Coding for Receiver-driven Layered Multicast. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.83.3540&rep=rep1&type=pdf>. Acesso em: 25 out. 2013.

[3]GIROD, B. Video Over Networks. Disponível em: <http://www.stanford.edu/class/ee398b/handouts/lectures/08-VideoOverNetworks.pdf>. Acesso em: 15 nov. 2012.

[4]AUSTERBERRY, D. The Technology of Video and Audio Streaming. Burlington: El Sevier, 2005, p. 133-243.

[5]JAY, M. The SHOUTcast Streaming Standard [Technical]. Disponível em: <http://forums.radiotoolbox.com/viewtopic.php?t=74>. Acesso em: 03 fev. 2013.

[6]ICECAST. Stream Structure per Mount Point. Disponível em: <http://www.icecast.org/docs.php>. Acesso em: 21 jan. 2013, il.

[7]REALNETWORKS. RealNetworks, Inc. History. Disponível em: <http://www.fundinguniverse.com/company-histories/realnetworks-inc-history/>. Acesso em: 25 out. 2013.

[8]HTML5. Thoughts on Flash. Disponível em: <http://www.apple.com/hotnews/thoughts-on-flash/>. Acesso em: 25 out. 2013.