Análise de segurança em servidores web usando HSTS

Julia Silva Weber, Ana Cristina Benso da Silva

Faculdade de Informática – Pontifícia Universidade Católica (PUC-RS)
Porto Alegre – RS – Brazil

julia.weber@acad.pucrs.br,ana.benso@pucrs.br

Abstract. *The HSTS protocol forces an HTTPS connection instead of HTTP. For this the HSTS policy imposes a severe security for both servers and browsers. Insecure connections are diverted to secure connections, and these must not degrade secure connections into insecure ones. This new mechanism, however, has not been fully proven in practice. The work seeks to test the effectiveness of HSTS through a series of tests in different scenarios, where the situation varies from client and server considering the security protocols used.*

Resumo. *O mecanismo HSTS visa forçar o uso de conexões HTTPS no lugar de HTTP. Para isso o HSTS impõe uma política de segurança severa tanto para servidores quanto para navegadores. Conexões inseguras devem ser desviadas para conexões seguras, e essas conexões seguras não devem degradar para conexões inseguras. Entretanto, o mecanismo ainda não foi amplamente adotado e necessita de comprovações a respeito de sua efetividade. Este trabalho busca testar a eficácia do HSTS através de uma série de testes em diferentes cenários, variando a configuração do cliente e servidor, os protocolos e mecanismos usados e também dos eventos gerados por possíveis intrusos.*

1 Introdução

Tendo em vista que a troca de dados é essencial, é preciso proteger as informações sensíveis que são transmitidas por redes de computadores. Os protocolos de comunicação, inicialmente, não tinham preocupação com segurança e, portanto, se mostraram vulneráveis a uma série de ataques. Posteriormente foram criados vários mecanismos, dentre eles o SSL (*Secure Socket Layer*)/TLS (*Transport Level Security*), que empregam técnicas de criptografia de chaves públicas para o estabelecimento de conexões. Entretanto a maioria dos usuários não utiliza conexões seguras e com isso fica vulnerável a ataques passivos como o de *sniffers* e ataques ativos como o homem do meio (Nakamura and Geus 2007). Em consequência desses ataques informações trocadas entre o cliente e o servidor podem ser roubadas por um atacante.

O HTTP (*HyperText Transfer Protocol*), um dos protocolos mais utilizados e que transporta muitas vezes dados sensíveis do usuário, é suscetível a ataques como homem do meio. Assim, para melhorar os aspectos de segurança deste protocolo, foi desenvolvido o mecanismo HSTS (*HTTP Strict Transport Security*) (J. Hodges, Jackson, and Barth 2012). Ele é utilizado para obrigar o usuário a usar o protocolo HTTPS (*HTTP Secure*) para acesso a um servidor, ao invés de usar conexões inseguras com HTTP. O HSTS é relativamente novo; em teoria, tudo indica que ele melhora a

segurança, mas na prática ainda é incerto se ele realmente só traz benefícios para o usuário.

O objetivo deste trabalho é avaliar, através de testes, o mecanismo HSTS para determinar quais poderiam ser seus pontos fortes e suas fragilidades. O comportamento do HSTS frente a ataques é analisado, observando se os problemas de segurança apontados na RFC 6797 (J. Hodges, Jackson, and Barth 2012) são realmente resolvidos.

Este trabalho está organizado como segue: na seção 2 é apresentado o referencial teórico, na seção 3 são detalhados os testes e na seção 4 é mostrada a conclusão.

2 HTTP Strict Transport Security (HSTS)

O HSTS é um mecanismo de segurança que permite somente que clientes, que se comunicam usando HTTPS, estabeleçam uma conexão com um servidor. Com este mecanismo ativo, os usuários que usam apenas o protocolo HTTP terão o pedido de conexão negado ao solicitar uma conexão com um servidor que utiliza o HSTS.

A razão de negar comunicações em HTTP é simples: se, hipoteticamente, o servidor web aceita uma conexão via HTTP e depois a redireciona para HTTPS, a troca de mensagens inicial será realizada sem criptografia, e nesse momento ataques oportunistas, como homem do meio, podem ocorrer. Essa brecha é suficiente para redirecionar o usuário para um *site* malicioso (Hodges, Jackson, and Barth 2012). Portanto, quando o servidor recebe um pedido de conexão usando HTTP, ele neste momento informa que para ter uma conexão com o servidor é necessário usar HTTPS. Dessa forma, todas tentativas seguintes de acesso ao *site* usando HTTP devem mudar para HTTPS. A aplicação desse mecanismo também obriga o navegador a armazenar uma lista contendo os *sites* que devem ser obrigatoriamente acessados via HTTPS.

2.1 Requisitos para uma conexão segura

No entanto, o HSTS somente tem o efeito desejado se sua implementação, em servidores e clientes, obedecer aos requisitos essenciais para que uma conexão segura seja corretamente estabelecida (J. Hodges, Jackson, and Barth 2012). Os requisitos são apresentados na Tabela 1.

Tabela 1 – Requisitos do HSTS

1	Servidores devem ser capazes de declarar para o navegador que os mesmos precisam usar uma política de segurança rígida.
2	Servidores devem ser capazes de instruir o navegador que requisita uma conexão insegura (HTTP) para fazê-lo de forma segura (com HTTPS).
3	Navegadores devem armazenar dados persistentes sobre os servidores que requisitam uma política de segurança rígida. Também devem permitir que esses servidores atualizem as suas políticas de segurança.
4	Navegadores precisam reescrever todos os acessos HTTP para utilizar o esquema HTTPS para aqueles servidores que utilizam política de segurança.
5	Administradores devem poder especificar políticas de segurança para subdomínios a partir de domínios de maior nível; os navegadores devem obedecer à especificação.
6	Navegadores não devem permitir que os subdomínios apliquem as políticas de segurança para domínios de maior nível.
7	Navegadores devem impedir que usuários ignorem avisos de segurança (*clicking through*). Não deve ser apresentada ao usuário a opção de continuar.

Se esses requisitos forem cumpridos, em teoria o usuário vai ter uma conexão mais segura, pois tanto servidor quanto navegador sempre darão prioridade ao uso de HTTPS e irão evitar usar o HTTP.

2.2 Mecanismo de segurança do HSTS

Um servidor declara a si mesmo como um servidor HSTS enviando uma Política HSTS para os navegadores. As políticas são informadas pelo envio de diretrizes para o cliente através do campo *Strict-Transport-Security HTTP response header field* (HSTS *header*), onde uma diretriz consiste de um nome e um valor. Atualmente estão definidas duas diretrizes: *Max-age* (quantos segundos o navegador deve considerar o servidor como um servidor HSTS) e *includeSubDomains* (especifica para o navegador qual a política que se aplica ao servidor e a qualquer subdomínio do domínio do servidor).

Para garantir a política de segurança, quando responder a uma requisição HTTP em um canal seguro, o servidor com HSTS deve incluir na sua resposta o HSTS *header*. Se receber uma requisição HTTP em um canal inseguro, ele deve enviar uma resposta HTTP contendo um código de redirecionamento permanente (código 301) para HTTPS. Um servidor HSTS não deve incluir um HSTS *header* em canais inseguros.

Quanto ao navegador, os requisitos descritos na Tabela 1 devem ser obedecidos. Quando preparar uma conexão HTTP para um servidor HSTS, o navegador deve: usar HTTPS no URL; substituir referências explícitas à porta 80 pela porta 443; preservar referências explícitas para quaisquer portas diferentes da 80; não incluir nenhuma porta, se não existir referência explícita. Outro aspecto a ser observado é a utilização de certificados auto-assinados. Se o cliente receber um certificado digital auto-assinado de um servidor HSTS, e o certificado raiz correspondente não está embutido no navegador ou no sistema operacional, então as conexões seguras para esse site devem falhar. Isso é feito para proteger contra os ataques ativos (como certificados falsos).

Além da negociação no momento do estabelecimento da conexão, outra forma proposta pelo HSTS de forçar o navegador a estabelecer conexões seguras, é utilizar no navegador uma lista pré-configurada de servidores HSTS conhecidos. Isso é útil para se proteger contra vulnerabilidades de homem do meio quando o HSTS é inicialmente ativado para cada servidor acessado.

Considera-se, na teoria, que esses mecanismos provêm a segurança necessária. Neste trabalho assume-se que o seu comportamento e a sua implementação devem ser testados para averiguar possíveis brechas de segurança.

2.3 HSTS: Ameaças

Segundo a RFC 6797 (Hodges, J., C. Jackson, and A. Barth 2012), o mecanismo de HSTS visa proteger as conexões HTTP de ataques clássicos como *sniffers* e homem do meio, porque o navegador deve usar HTTPS e é obrigado a interromper a conexão se um *site* não puder ser autenticado de forma confiável.

Porém, há brecha para a ocorrência de um ataque caso seja permitido o *clicking through*. Por exemplo, um atacante pode explorar um erro simples que é ter, em um *site*, referências inseguras a conteúdos de outros *sites*. Nessa brecha, o atacante pode interferir na conexão segura através dessas conexões inseguras. Se o desenvolvedor do *site* evitasse *links* não confiáveis, o problema não existiria.

O HSTS não foi projetado para proteger de todos os tipos de ameaças. Por exemplo, o HSTS, pelas suas caraterísticas, não trata *phishing*. O HSTS, entretanto, complementa muitas defesas de *phishing* existentes, instruindo o navegador a proteger a integridade da sessão e dos *tokens* de autenticação de longa duração (Jackson and Barth 2008). O HSTS também não lida com vírus e semelhantes. O HSTS também não age como um *firewall*, ele não tem funções para conseguir detectar tráfego malicioso.

Ainda o mecanismo poderá estar sujeito a ameaças decorrentes da sua forma de funcionamento. Atacantes que executam um ataque *bootstrap man-in-the-middle*, ou seja, no momento anterior ao estabelecimento da conexão segura, poderão explorar diversos aspectos tais como criar situações de negação de serviço, obrigando o uso de conexões seguras para sites que não as utilizam, ou tentar mascarar o servidor para alterar informações do atributo *max-age*, a fim de desabilitar o mecanismo, ou também tentar promover a alteração da lista de *sites* pré-configurados mantida pelos navegadores (Sugavanesh, Hari Prasath, and Selvakumar 2013).

3 Teste de segurança do HSTS

Nesse artigo, são apresentados os resultados dos testes realizados sobre possíveis ameaças que poderiam comprometer uma conexão HTTP. O objetivo deste trabalho é verificar a eficácia do HSTS em conexões entre servidores e navegadores, sendo assim a metodologia de teste que melhor se aplica é a de Auditoria de Segurança (Bishop 2003). Assume-se que o sistema operacional e os aplicativos dos servidores e navegadores estão livres de vulnerabilidades conhecidas e, portanto, não se aplicam testes do tipo descoberta, varredura e detecção de vulnerabilidades (Bishop 2003). Teste de penetração (Bechtsoudis and Sklavos 2012) não é o mais adequado, pois não se deseja testar falhas de segurança, mas sim verificar a efetividade do HSTS. Uma auditoria de segurança (*Security Audit)* é uma avaliação técnica de um sistema ou aplicativo. Normalmente a auditoria é motivada pela análise de um controle específico ou verificação de obediência de uma norma.

3.1 Cenários e Resultados

Para os testes são necessários três computadores distintos ligados em rede. Um faz o papel de **Servidor** web, com SO Linux e servidor Apache. Outro faz o papel de **Cliente**, com Windows e executa os navegadores Chrome, Firefox, Opera e Internet Explorer. O último computador faz simultaneamente os papéis de **Atacante**, tanto ativo como passivo, e de auditor, registrando todas as mensagens trocadas entre **Cliente** e **Servidor**. Neste computador é executado o analisador de tráfego Wireshark [http://www.wireshark.org/], e o seu SO pode ser tanto Linux quanto Windows, pois o importante é a ferramenta e não o seu sistema operacional. Essas máquinas rodam em um sistema virtual, usando VirtualBox [https://www. virtualbox.org/], em um computador com suporte a virtualização em hardware. Isso permite manter o ambiente compacto para a realização dos testes. Para a conexão das três máquinas é definida uma rede local.

No **Atacante**, ataques passivos são realizados com o Wireshark. Ataques ativos envolvem um ataque do homem-do-meio, que é implementado alterando a *cache* de *hosts* do Cliente para simular um ataque de envenenamento de *cache*. Foi gerado um certificado raiz (auto-assinado) para simular uma Autoridade Certificadora e um certificado para o **Servidor**, devidamente assinado pela Autoridade Certificadora. Para o

cliente reconhecer esses certificados como válidos, o certificado raiz foi instalado no **Cliente**. No **Atacante** foram gerados outros certificados, não assinados pelo certificado raiz, para serem usados como certificados falsos nos ataques de homem do meio. Para geração destes certificados, foi utilizado o OpenSSL.

3.2 Resultados obtidos

Para a metodologia de testes foram criados 21 cenários, envolvendo servidor e cliente sem suporte, HSTS somente no servidor, HSTS somente no cliente e HSTS em ambos. Ataques variavam entre ataques passivos, onde era realizado somente um *sniffer* através do *Wireshark* e ativos, onde era realizado um ataque do homem do meio através de *ARP spoofing*, envenenamento de DNS e SSL *striping* usando o programa *ettercap*.

Nos testes em que tanto servidor quanto cliente não utilizaram HSTS, todos os dados da conexão foram facilmente roubados pelo atacante. Por outro lado, as conexões que apresentam o mecanismo HSTS, tanto no navegador quanto no servidor, impediram a realização dos ataques. É possível observar que no conjunto de testes em que somente o navegador possui HSTS e o servidor não tenha esse mecanismo, os dados são roubados com a mesma facilidade do que nos casos de conexões HTTP. Portanto é possível concluir que navegadores com suporte a HSTS não contribuem para o aumento da segurança quando acessam um servidor que não utiliza o mesmo mecanismo. Isso apresenta problemas para casos de *clicking through*, onde o usuário é redirecionado para sites que não utilizem HSTS. Nesses casos, mesmo que em primeira instância o cliente tenha realizado uma conexão segura, no momento em que o ele for redirecionado a outro servidor, ele estará novamente a mercê de ameaças. Para amenizar estes problemas, atualmente alguns navegadores, como Google Chrome e Firefox, tem pré-carregada uma lista de sites que obrigatoriamente devem ter HSTS habilitado [http://www.chromium.org/sts].

Os servidores que suportam HSTS instruem os navegadores a só se conectar através de HTTPS por um período de tempo (*max age*). Para evitar uso indevido, a ativação do HSTS só pode ser realizada através de HTTPS, e nunca através de HTTP. Conexões HTTPS cifram os dados com criptografia forte e permitem autenticação do servidor através de certificados digitais, minimizando assim as chances de um atacante obter sucesso. Entretanto, nos testes em que o servidor utilizava HSTS, mas o navegador não, a proteção era parcial. O servidor redirecionava conexões HTTP para HTTPS, mas nos acessos seguintes o navegador voltava a utilizar HTTP e, portanto, voltava a ficar vulnerável. Com suporte a HSTS, um navegador somente volta a fazer conexões através de HTTPS.

No momento da escrita deste artigo, existia suporte a HSTS nos navegadores Google Chrome e Chromium, Firefox, Opera, Safari e Seamonkey. A versão atual do Internet Explorer (11.0) ainda não oferece suporte, mas a próxima versão (12.0) promete incluir este suporte. Em termos de servidores, há suporte para inclusão de HSTS no Apache, *lighttpd* e *nginx*. Para o *Internet Information Services* (IIS) da Microsoft ainda não existe suporte oficial, mas existem módulos de domínio público que realizam essa tarefa.

4 Conclusão

O mecanismo HSTS impõe o uso de conexões seguras entre cliente e servidor, visando evitar a ocorrência de três tipos de ameaças. A primeira ameaça ocorre quando um

navegador solicita conexão HTTP a um servidor que utiliza HTTPS. Nesse caso o HSTS força um redirecionamento para uma conexão HTTPS, impedindo ataques do homem do meio em conexões inseguras. A segunda ocorre quando existem *links* inseguros (por HTTP) para sites que usam HSTS. Aqui também as conexões inseguras são redirecionadas para HTTPS. E a terceira ameaça ocorre no caso de certificados digitais falsos ou inválidos. HSTS obriga o navegador a emitir uma mensagem de erro no caso de certificados auto-assinados, assinados por uma entidade desconhecida, fora do prazo de validade ou utilizados para o endereço (URL) incorreto. Além disso, o navegador não deve oferecer ao usuário a opção de ignorar o erro. Com base no estudo realizado, observa-se que o HSTS é de fato efetivo, quanto utilizado tanto no servidor como no navegador, e seu uso impede grande parte dos problemas que existem atualmente me conexões web.

Entretanto, a adoção do HSTS ainda não está generalizada. Muitos sites não configuraram seus servidores para suportar HSTS, apesar da grande maioria dos navegadores já oferecer este suporte. Adicionalmente, existe a vulnerabilidade do *Boostrap MITM*, onde um ataque no homem do meio é realizado antes do primeiro acesso a um servidor que use HSTS, e, portanto, antes da conexão se tornar segura. Para minimizar este problema, utiliza-se a lista pré-carregada de sites que o navegador deve obrigatoriamente acessar por HTTPS. Curiosamente, segundo a RFC6797, o HSTS poderia ser usado para realizar um ataque de negação de serviço, quando um atacante consegue injetar no navegador uma política HSTS para um servidor que não tem suporte a HTTPS. O navegador vai tentar se conectar de forma segura, mas este serviço não está disponível, e assim o site fica inacessível Entretanto, nos testes realizados isto não ocorreu, pois o navegador não aceitou a política HSTS do atacante porque detectou que o Certificado do atacante era inválido. Apesar destas situações, a adoção do HSTS reforça em muito a segurança, pois forçam navegadores a somente realizar acesso por HTTPS. Mas a existência dos ataques citados demostram que o mecanismo HSTS ainda possui aspectos que devem ser melhorados.

5 Referências bibliográficas

Bechtsoudis, Anestis, and Nicolas Sklavos. 2012. “Aiming at Higher Network Security through Extensive Penetration Tests.” *Latin America Transactions, IEEE (Revista IEEE America Latina)* 10 (3): 1752–56.

Bishop, Matt. 2003. *Computer Security: Art and Science*. Addison-Wesley Professional.

Hodges, J., C. Jackson, and A. Barth. 2012. “HTTP Strict Transport Security (HSTS),” November. http://www.rfc-editor.org/info/rfc6797.

Jackson, Collin, and Adam Barth. 2008. “Forcehttps: Protecting High-Security Web Sites from Network Attacks.” In *Proceedings of the 17th International Conference on World Wide Web*, 525–34. ACM.

Nakamura, Emilio Tissato, and Paulo Lício Geus. 2007. *Segurança de Redes Em Ambientes Corporativos*. São Paulo: Novatec.

Sugavanesh, B, R. Hari Prasath, and S Selvakumar. 2013. “SHS-HTTPS Enforcer: Enforcing HTTPS and Preventing MITM Attacks.” *SIGSOFT Softw. Eng. Notes* 38 (6): 1–4. doi:10.1145/2532780.2532802.