Análise de Segurança da Camada de Notificação do SparkleShare

Nícolas Kaqui, Marcelo Akira, Bruno Silvestre

[1]Instituto de Informática — UFG

{nicolaskaqui, marceloakira, brunoos}@inf.ufg.br

Abstract. *In this work, we analyzed SparkleShare project about some security questions. This project is an alternative for Dropbox or UbuntuOne. We identified some security threats in one component of SparkleShare and we proposed a solution to mitigate the problem.*

Resumo. *Neste trabalho, analisamos quesitos de segurança do projeto SparkleShare, que é uma solução alternativa ao Dropbox ou UbuntuOne para sincronização automatizada de arquivos. Identificamos algumas falhas de segurança em um dos componentes do SparkleShare e propomos uma solução para mitigar o problema.*

1. Introdução

O grande desenvolvimento da sociedade fez com que os serviços básicos e essenciais sejam quase todos entregues de uma forma completamente transparente. As infraestruturas existentes permitem entregar os serviços em qualquer lugar e a qualquer hora, de modo que possamos acender a luz, abrir a torneira ou fazer uma ligação para qualquer lugar. Esse mesmo conceito tem sido aplicado na informática e uma mudança consistente tem sido feita com a disseminação de um novo paradigma, a computação em nuvem.

A computação em nuvem [Taurion 2009, Armbrust et al. 2010] é um paradigma que muda o modelo de computação tradicional, onde todo o processamento de dados e o armazenamento de arquivos é feito em um computador local, para um modelo onde essas tarefas são realizadas por um ou vários computadores na Internet. Esse paradigma pretende ser global e prover serviços para todos, desde o usuário final que hospeda seus documentos pessoais na Internet, até empresas que terceirizam toda a parte de TI.

Com a computação em nuvem, os usuários moverão seus dados e aplicações para a nuvem, podendo acessá-los de forma simples e de qualquer local [Drago et al. 2012]. Sendo uma ferramenta de trabalho importante em ambientes onde ocorre uma troca de arquivos entre vários usuários e para usuários que necessitam acessar seus arquivos de diferentes lugares, como por exemplo, nas empresas, nos escritórios e nas escolas. Os sistemas que permitem a utilização desses documentos em diversos lugares geralmente empregam mecanismos de sincronização para manter a consistência [Xianqiang et al. 2011].

O SparkleShare é um sistema que permite ao usuário manter seus arquivos sincronizados entre várias máquinas. Diferentemente do Dropbox [Dropbox Inc. 2012, Drago et al. 2012] ou UbuntuOne [Canonical Ltd. 2012], cujos serviços são oferecidos por empresas, o SparkleShare é uma solução independente e qualquer instituição ou usuário final podem utilizá-lo para montar seus serviços de sincronização de arquivos.

Em nossos estudos sobre o funcionamento do SparkleShare (versão 0.4.0), identificamos problemas de segurança em seu mecanismo de notificação de sincronização. Neste trabalho, apresentamos a análise feita e como foi realizada a correção do problema.

O trabalho está estruturado da seguinte forma: na seção 2 apresentamos a arquitetura interna do SparkleShare, na seção 3 apresentamos o problema de segurança identificado e a solução adotada, e na seção 4 apresentamos a conclusão do trabalho.

2. SparkleShare

O SparkleShare surgiu como um projeto que permitisse que usuários e empresas pudessem criar o seu próprio repositório de sincronização automatizada de arquivos [Bons 2010].

A possibilidade de trabalhar com mais de um repositório de arquivos e cada repositório estar hospedado em um servidor diferente fez com que o SparkleShare fosse bem visto por empresas e usuários que já possuem um ou mais repositórios e necessitam do sistemas para sincronizar e compartilhar os arquivos [Bons 2012].

O cliente SparkleShare (instalado na máquina do usuário) é composto pelas camadas de sincronização e notificação. A camada de sincronização utiliza um repositório Git para armazenar e manter o histórico dos arquivos, e a camada de notificação utiliza um serviço IRC para troca de mensagens. A Figura 1 apresenta a arquitetura em mais detalhes.

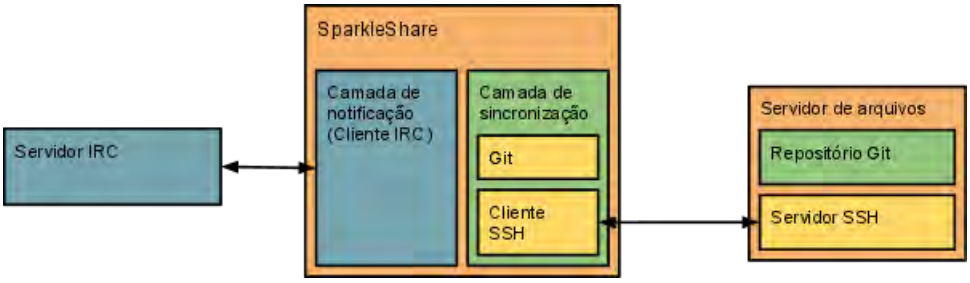

Figura 1. Apresentação das camadas na arquitetura do SparkleShare

A seguir, faremos uma análise das camadas de sincronização e notificação. Apresentaremos o funcionamento e a estrutura de cada uma delas.

2.1. Camada de Sincronização

A camada de sincronização é responsável pelo armazenamento, versionamento e sincronização dos arquivos no repositório central (servidor). O funcionamento dessa camada ocorre em dois momentos: quando o repositório local é modificado ou quando recebe uma notificação remota de alteração do repositório.

Quando o repositório local sofre alguma modificação (geralmente, o usuário modificando seus arquivos), o mecanismo de sincronização registra as modificações no servidor e notifica a mudança pela camada de notificação. Quando o cliente SparkleShare recebe uma notificação de atualização, a camada de sincronização é ativada para que se inicie a sincronização dos arquivos com o repositório central.

A camada de sincronização é implementada utilizando o sistema de controle de versão Git [Loeliger 2009]. Esse sistema foi idealizado e desenvolvido por Linus Torvalds, tendo como principal características a licença livre e o suporte ao desenvolvimento distribuído.

Um conceito importante (e usado na camada de notificação do SparkleShare) é o identificador de *commit* usado pelo Git. Cada versão tem um identificador produzido pela aplicação do algoritmo SHA1 ao conteúdo modificado, produzindo assim um valor de *hash*, por exemplo:

```
68bd599b85538ae08c7729ce8d5127076f567b29
```

O objetivo é que essas referências sejam únicas no repositório e possam ser usadas como referência para o *commit* [Swicegood 2008].

O Git oferece suporte de acesso ao repositório por HTTP, SSH ou protocolo próprio (Git). A versão analisada do SparkleShare utiliza o Git com a comunicação sendo feita pelo SSH, garantindo autenticação do usuário no servidor e canal seguro de comunicação.

2.2. Camada de Notificação

A camada de notificação é utilizada como meio de comunicação entre os clientes SparkleShare. É por essa camada que um cliente avisa aos demais clientes do usuário que é necessário atualizar os arquivos pois foi feita uma modificação no repositório.

A camada é implementada utilizando o protocolo IRC [Charalabidis 1999]. A escolha do protocolo IRC ocorreu pela facilidade de configuração e o baixo *overhead*. Por ser um protocolo de comunicação simples e focado em troca de mensagens simples (apenas texto), o IRC, na época em que foi desenvolvido, se destacou muito pela agilidade de comunicação entre os clientes, mesmo sobre uma conexão considerada hoje lenta [Toyer and Smith 1997].

Dessa forma, o SparkleShare utiliza um servidor IRC como sendo um serviço *Publish-Subscribe*. Os clientes de um usuário se formam uma rede ponto-a-ponto e se conectam ao mesmo servidor IRC para acessar um canal em comum e trocar informações. Assim, o objetivo é que a troca de notificações só ocorra entre eles.

A definição do canal de acesso comum é dado pelo primeiro identificador de *commit* do repositório Git ao qual o cliente faz referência. Podemos ter então canais como a *hash* mostrada na seção 2.1. O uso desse identificador minimiza a chance de colisão no nome dos canais.

Com o objetivo de que fosse fácil para os usuários, em termos de infraestrutura, utilizarem o SparkleShare, o projeto disponibilizou um servidor IRC público de livre acesso, que é usado por padrão. Caso o usuário desejar mudar, basta alterar o arquivo de configuração do SparkleShare.

3. Segurança no SparkleShare

Após a análise mais interna das camadas do SparkleShare, pudemos identificar problemas de segurança quanto ao protocolo utilizado na camada de notificação (IRC).

Por ser de simples implementação, o protocolo IRC não possui uma boa camada de segurança, podendo ser utilizada como meio de possíveis ataques. Por esse motivo, é muito comum ver o protocolo IRC bloqueado por *firewall* em muitos ambientes [Toyer and Smith 1997].

Na construção da camada de notificação do SparkleShare, algumas medidas de segurança do IRC não foram utilizadas, sujeitando o usuário do sistema a alguns possíveis ataques, por exemplo, o ataque de *footprinting* [Assunção 2010]. Nesse tipo de ataque são coletadas informações do usuário, como IP da máquina e canais que ele está utilizando. Usuários que optaram por um servidor IRC privado podem ter o problema reduzido, mas os usuários que fazem uso do servidor público do projeto são mais afetados, devido à exposição.

Utilizando um cliente IRC comum, conectamos no servidor IRC público e realizamos apenas a listagem dos canais existentes, para ilustrar. A figura 2 mostrado a listagem realizada.

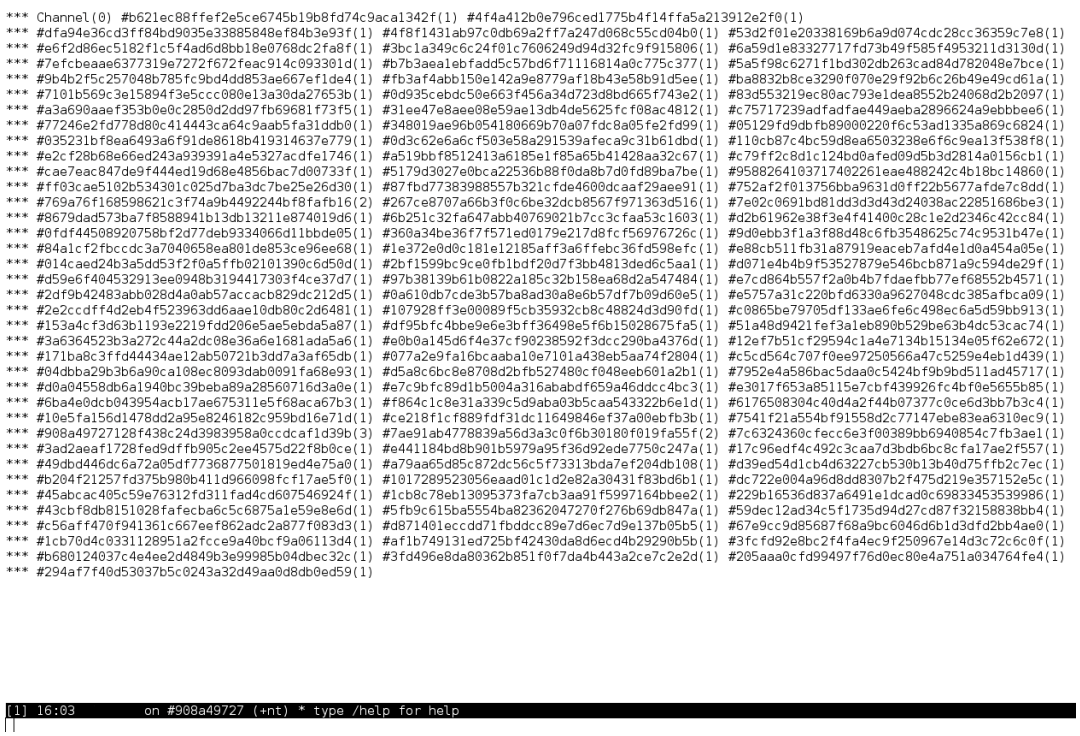

Figura 2. Listagem dos canais no servidor IRC do SparkleShare.

Além disso, os canais criados pelo SparkleShare não possuem nenhuma proteção. Para exemplificar, como mostrado na Figura 3, utilizando novamente um cliente IRC comum, conseguimos acessar o canal de nossa instalação do SparkleShare. Vários autores citam que a segurança de sistemas baseados em publish/-subscribe ou P2P começa com a garantia de que não há vazamento ou infiltração de mensagens externas [Bacon et al. 2008, Singh et al. 2011, Bailes and Templeton 2004, Zhang et al. 2005]. Pode-se notar que essa premissa foi claramente violado na implementação do SparkleShare que analisamos.

Como o cliente SparkleShare não possui nenhum formato para a mensagem de

Figura 3. Acessando um canal IRC do SparkleShare por cliente IRC externo

notificação, qualquer mensagem postada no canal (até mesmo uma simples letra) é interpretada como uma indicação de que o repositório foi alterado. Somando isso ao acesso aberto aos canais, é possível enviar falsas notificações.

Nesse caso, é possível construir um pequeno programa que varre todo os canais do IRC enviando uma mensagem qualquer. Isso fará todos os clientes irem ao repositório central, ao mesmo tempo e repetidamente, para realizar a atualização da base local. Esse volume de requisições pode causar desde um transtorno até tornar o servidor inacessível — causando o chamado *Denial of Service* [Assunção 2010].

3.1. Soluções de Segurança

Tendo identificado algumas vulnerabilidades do SparkleShare 0.4.0, foram estudadas algumas soluções de segurança que tornam a camada de notificação mais segura. As soluções propostas para garantir a segurança dos usuários e dos canais dentro do IRC são: canais no modo secreto e canais com senha.

A especificação do IRC permite que o usuário configure um canal em modo secreto. Esse tipo de canal não poderá ser listado pelos usuários, dessa maneira é necessário que o usuário do canal saiba o nome do canal para poder acessá-lo [Charalabidis 1999].

Essa solução contribui para reduzir as informações disponíveis, e como o *hash* é utilizado como nome do canal, isso oferece um nível maior de segurança. Por outro lado, não afeta o funcionamento dos clientes, dado que eles possuem o identificador do repositório Git.

Entretanto, a abordagem descrita não protege do fato do *hash* for conhecido de antemão (de acessos inseguros anteriores) ou gerado (pois é apenas uma string). E como soluções de segurança não podem ser simplesmente baseadas na ocultação de informações, a segunda medida que propomos é o uso de senha para acesso ao canal oferecido pelo protocolo IRC.

Definimos que os clientes SparkleShare ligados a um mesmo repositório Git sejam configurados com uma senha. O primeiro cliente a se conectar com o servidor IRC cria o canal especificando a senha compartilhada, os demais necessitam da senha para fazer parte do canal.

Mas temos que considerar ainda que o canal pode já existir e ele não estar protegido por senha, o que leva a dois casos: (i) um cliente sem suporte à senha foi ativado primeiro ou (ii) alguém criou o canal para esperar conexões de clientes e aplicar algum ataque.

Considerando a possibilidade do caso (ii), a melhor abordagem seria impedir que o cliente se conectasse com um canal não protegido. Mas, os clientes SparkleShare de um usuário podem estar instalados em diversas máquinas e o processo de atualização não é feito de forma instantânea. Dessa forma, os novos clientes atualizados para a nossa versão não poderiam funcionar até todos os clientes estiverem atualizados — o que pode gerar um transtorno para o usuário.

Para se acomodar a esse período de transição, optamos então por uma solução em que o usuário pode escolher se o cliente deve ou não se conectar a um canal não protegido por senha. Dessa forma, o usuário poderá atualizar os clientes sem necessariamente impedi-los de funcionar. Quando todos os clientes estiverem atualizados, o usuário poderá então ativar a proteção.

Novos usuários não terão o problema de atualização e poderão iniciar já com a proteção de canais ativada.

3.2. Comentários sobre a Implementação das Soluções

Durante o processo de aplicação das modificações no código fonte do SparkleShare, nos deparamos com o fato da solução dos canais em modo secreto já ter sido implementada na versão de desenvolvimento [Bons 2011]. Focamos então na implementação dos canais com senha.

O SparkleShare é desenvolvido em C# e possui uma classe (`SparkleListenerIrc.cs`) que realiza todo o processo da comunicação do sistema com o cliente IRC. Essa classe por sua vez utiliza a biblioteca SmartIrc4net [Bauer 2012]. Modificamos então o trecho de conexão no canal, introduzindo o modo protegido por senha.

A configuração da senha e das políticas de acesso (forçar o acesso somente por senha ou não) aos canais é feita no arquivo de configuração XML padrão do SparkleShare. No sistemas operacional Debian, onde os testes foram realizados, o arquivo de configuração é o seguinte:

```
/home/user/.config/sparkleshare/config.xml
```

O usuário deve incluir as seguintes propriedades nesse arquivo de configuração para que a proteção funcione adequadamente:

- **announcements_password**: senha do canal.
- **allow_passwordless_join**: booleano que indica a permissão para acessar canais não protegidos.

As modificações foram testadas com um repositório no GitHub [GitHub Inc. 2011] e com o servidor público IRC do SparkleShare. Depois dos testes, um *patch* foi oferecido aos desenvolvedores do SparkleShare, que incorporaram as nossas modificações no repositório do projeto [Kaqui 2011]

4. Considerações Finais

Com a evolução da computação em nuvem e a popularização dos serviços oferecidos por ela, foram desenvolvidos os sistemas de sincronização e compartilhamento de arquivos com a finalidade de abstrair a utilização do armazenamento de arquivos em nuvem e de oferecer um serviço de sincronização de arquivos locais com a nuvem. O SparkleShare é um sistema de sincronização desenvolvido dentro da filosofia de Software Livre e permite que os usuários e instituições utilizem diferentes repositórios de armazenamento.

Por utilizar elementos disponíveis na Internet, o funcionamento do SparkleShare está sujeito a ataques. Neste trabalho realizamos uma análise de segurança do SparkleShare, versão 0.4.0, identificamos alguns pontos falhos em seu serviço de notificação, e aplicações soluções que aumentam a segurança.

Os desenvolvedores do sistema já reconheceram que o uso do IRC não é uma boa opção, sendo que na época em que este trabalho foi realizado, o desejo dos desenvolvedores era substituir pelo protocolo XMPP [Foundation 2004]. Durante a escrita deste artigo, verificamos que o projeto agora utiliza uma solução própria para notificação.

Ainda assim, acreditamos que a nossa solução de canais protegidos por senha implementada era de importância para o projeto, pois oferecia um nível melhor de segurança para os usuários, mantendo compatibilidade com versões anteriores e de baixo impacto no código (comparado com adaptar a camada de notificação para um novo protocolo).

Destacamos novamente que essa melhoria foi enviada aos desenvolvedores do SparkleShare e incluída no código final do sistema.

Referências

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53:50–58.

Assunção, M. (2010). *Guia do Hacker Brasileiro*. VisualBooks.

Bacon, J., Eyers, D. M., Singh, J., and Pietzuch, P. R. (2008). Access control in publish/-subscribe systems. In *Proceedings of the Second International Conference on Distributed Event-based Systems*, DEBS '08, pages 23–34, New York, NY, USA. ACM.

Bailes, J. E. and Templeton, G. F. (2004). Managing P2P security. *Commun. ACM*, 47(9):95–98.

Bauer, M. (2012). SmartIrc4net - C# IRC Library. `http://www.meebey.net/projects/smartirc4net/`.

Bons, H. (2010). Announcing SparkleShare. Design Monkey. `http://www.bomahy.nl/hylke/blog/announcing-sparkleshare/`.

Bons, H. (2011). GitHub commit description. `https://github.com/hbons/SparkleShare/commit/5f31d5f43d41a34300eee886bef9bc8e1a52800c`.

Bons, H. (2012). SparkleShare wiki. `https://github.com/hbons/SparkleShare/wiki`.

Canonical Ltd. (2012). Ubuntu One. http://one.ubuntu.com.

Charalabidis, A. (1999). *Book of IRC: The Ultimate Guide to Internet Relay Chat*. The Book Of Series. No Starch Press.

Drago, I., Mellia, M., M. Munafo, M., Sperotto, A., Sadre, R., and Pras, A. (2012). Inside Dropbox: Understanding personal cloud storage services. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC '12, pages 481–494, New York, NY, USA. ACM.

Dropbox Inc. (2012). Dropbox. https://www.dropbox.com/.

Foundation, J. S. (2004). Extensible messaging and presence protocol (XMPP): Core. RFC 3920.

GitHub Inc. (2011). GitHub. https://github.com/.

Kaqui, N. (2011). GitHub commit description. https://github.com/hbons/SparkleShare/commits/0.4.0/SparkleLib/SparkleListenerIrc.cs?author=nicolaslazartekaqui.

Loeliger, J. (2009). *Version control with Git*. O'Reilly Series. O'Reilly.

Singh, J., Eyers, D. M., and Bacon, J. (2011). Disclosure control in multi-domain publish/subscribe systems. In *Proceedings of the 5th ACM International Conference on Distributed event-based system*, DEBS '11, pages 159–170, New York, NY, USA. ACM.

Swicegood, T. (2008). *Pragmatic Version Control Using Git*. Number v. 1 in Pragmatic Version Control Using Git. Pragmatic Bookshelf.

Taurion, C. (2009). *Cloud Computing - Computação em Nuvem: Transformando o Mundo da Tecnologia da Informação*. BRASPORT.

Toyer, K. and Smith, N. (1997). *Learn advanced Internet relay chat*. Wordware Pub.

Xianqiang, B., Nong, X., Weisong, S., Fang, L., Huajian, M., and Hang, Z. (2011). Syncviews: Toward consistent user views in cloud-based file synchronization services. In *Chinagrid Conference (ChinaGrid), 2011 Sixth Annual*, pages 89–96.

Zhang, X., Chen, S., and Sandhu, R. (2005). Enhancing data authenticity and integrity in P2P systems. *Internet Computing, IEEE*, 9(6):42–49.