Curso de Sistemas de Informação

Universidade Estadual de Mato Grosso do Sul

AutoSSH: automação da conexão SSH da ferramenta SVarLog do SAESD

Alessandro Dutra de Andrade

Dr. Ricardo Luís Lachi (Orientador)

Dourados - MS
2016

Coordenação do Curso de Sistemas de Informação
Universidade Estadual de Mato Grosso do Sul

AutoSSH: automação da conexão SSH da ferramenta SVarLog do SAESD

Alessandro Dutra de Andrade

Novembro de 2016

Banca Examinadora:

- Prof. Dr. Ricardo Luís Lachi (Orientador)
- Prof. Dr. Fabrício Sérgio de Paula
- Prof. Dr. Nilton Cézar de Paula

AutoSSH: automação da conexão SSH da ferramenta SVarLog do SAESD

Alessandro Dutra de Andrade

Projeto de pesquisa apresentado a disciplina de Projeto Final de Curso II, no curso de Sistemas de Informação da Universidade Estadual de Mato Grosso do Sul, ano letivo de 2016, como parte dos requisitos para obtenção do título de Bacharel em Sistemas de Informação.

Dourados, 10 de novembro de 2016.

Dr. Ricardo Luís Lachi (Orientador)

Agradecimentos

Primeiramente agradeço a Deus, por dar força e perseverança para conclusão deste trabalho.

Agradeço aos meus pais Carlos e Ângela e meus familiares que sempre me apoiaram e acreditaram na minha capacidade.

A minha vó que me apoiou e ofereceu moradia para superar os momentos difíceis.

A minha noiva Renata, pelo apoio e por me fazer acreditar que tudo que temos e somos não é nada comparado ao amor que temos um pelo outro.

Ao meu orientador Professor Dr. Ricardo Luís Lachi pela orientação deste trabalho e que nunca me deixou desamparado e sempre acreditou que conseguiria concluir este trabalho.

A todos meus familiares e amigos que me ajudaram, de forma direta ou indiretamente, durante todos esses anos, na conclusão de mais esta etapa em minha vida.

Resumo

Neste trabalho foi construída uma ferramenta, denominada AutoSSH, que permite automatizar a parte de conexão Secure Shell (SSH) necessária para verificação de períodos de instabilidade de Sistemas Operacionais (SOs) baseados no Unix do Sistema de apoio a Avaliação de cursos do Ensino Superior a Distância (SAESD). O SAESD é um sistema que foi desenvolvido para dar suporte e facilitar a aplicação do modelo de avaliação que se baseia em questionários e é capaz de recuperar informações relevantes para avaliação da qualidade de um curso *online*, abrangendo desde a análise de *logs* do Sistema Operacional até o padrão de acessos dos participantes do curso *online*.

A ferramenta de varredura de *logs*, SvarLog um dos componentes do SAESD, precisa acessar remotamente o servidor de um sistema de Ensino a Distância (EaD) para analise de *logs* do SO Linux, este trabalho tem por objetivo automatizar a conexão do protocolo SSH, eliminando a necessidade de digitação de senha a cada conexão ou comando executado. O AutoSSH utiliza o conceito de chaves assimétricas, fazendo com que os computadores envolvidos no processo se autentiquem automaticamente. Além disso, a ferramenta permite a automatização do protocolo SSH considerando as versões atuais dele, que são: SSH-1, SSH-2 e OpenSSH.

Palavras chaves: AutoSSH, *Iogin* automático, Chaves assimétricas.

Abstract

In this work was built a tool called AutoSSH, it allows automate the part of Secure Shell (SSH) connection required to check unstable periods of Operating Systems (OS), these systems are based in the Unix of Support System Higher Education courses Rating Distance (SAESD). The SAESD is a system developed to support and facilitate the application of evaluation model, which is based on questionnaires and is able to retrieve relevant information to evaluate the quality of an online course, that include from the analysis of logs of Operating System to the standard access of the participants of the online course.

The log's scanning tool, SvarLog a component of SAESD, need to remotely access the server from a system of Distance Learning to analyse the SO Linux's logs, the objective of this work is automate the SSH connection pulling out the need of password for each connection or command executed. The AutoSSH uses the concept of asymmetric keys, causing the computers involved in the process authenticate itself automatically. Additionally, the tool allow the automation of the SSH protocol, considering the current versions of the protocol, that they are SSH-1, SSH-2 and OpenSSH.

Keywords: *AutoSSH, Automatic Login, asymmetric keys.*

Sumário

Lista de abreviaturas e siglas

IP	*Internet Protocol*
MSDN	*Microsoft Developer Network*
SO	Sistema Operacional
SAESD	Sistema de apoio a Avaliação de cursos do Ensino Superior a Distância
SSH	*Secure Shell*

Lista de figuras

Lista de tabelas

Capítulo 1

Introdução

O Sistema de apoio a Avaliação de cursos do Ensino Superior a Distância (SAESD) foi criado com a finalidade de avaliar a qualidade de cursos superiores a Distância (EaD). Essa busca pela qualidade se justifica, pelo fato da crescente procura pela modalidade de cursos EaDs, sendo assim, é de fundamental importância a existência de mecanismos computacionais para mensurar a qualidade destes cursos, afim de demostrar que esses têm a mesma excelência dos cursos superiores presenciais.

Segundo Lachi (2012), o SAESD reúne mecanismos para uma completa avaliação, contendo questionários e ferramentas de apoio para esse fim. Uma das ferramentas componentes do SAESD é a ferramenta de varredura de *logs* denominada SVarLog. Essa ferramenta compila a partir de uma análise dos *logs* do sistema operacional, informações sobre os *logins* feitos pelos usuários, e os períodos de desligamentos do Sistema, entre outras. Uma das características da ferramenta SVarLog é a necessidade de acesso remoto ao servidor de onde obterá os *logs* de sistema que irá analisar. Para esse acesso remoto o SVarLog faz uso do protocolo SSH.

Segundo Ylonen (2006), a principal característica do protocolo *Secure Shell* é fato dele ser um protocolo seguro, uma vez que os dados que trafegam pela rede por meio desse protocolo são criptografados. Outro aspecto relevante é que o SSH[1] vem por padrão em diversas distribuições Unix e existem também clientes SSH para as mais diversas plataformas, como por exemplo, para a plataforma do SO Windows.

Além disso, na versão atual, o SVarLog necessita que o usuário configure manualmente a conexão SSH para acesso remoto de modo que os *logs* possam ser coletados.

[1] Atualmente há diversos programas que utilizam este protocolo, nos sistemas derivados do Unix, tais como, SO Linux, OpenBSD, etc. Normalmente o SSH é instalado como parte da instalação padrão nesses SOs, já nos SOs Windows é necessário instalar clientes SSH.

Assim este trabalho teve como objetivo principal a automação do acesso remoto sem a necessidade de configuração manual do acesso remoto a um servidor Unix a partir de um SO Windows. É importante ressaltar que apesar do cenário ser específico para acesso remoto de um servidor baseado em Unix contendo o SSH servidor, através de um cliente SSH instalado no SO Windows, ele não implica na impossibilidade de seu uso e/ou adaptação para outros cenários em que haja um SO Windows comunicando com qualquer outro SO baseado no Unix.

Essa ferramenta permite ao usuário configurar a conexão através do protocolo SSH, podendo escolher previamente qual versão do protocolo deseja trabalhar e assim realizar a automatização da conexão da mesma, levando em conta as versões atuais do protocolo que são: SSH-1, SSH-2 e OpenSSH.

A organização do texto esta disposta da seguinte forma, o **Capítulo 2** apresenta uma breve explicação sobre o SAESD e a ferramenta SVarLog e explica o que foi automatizado através da ferramenta AutoSSH. **Capítulo 3** apresenta as funcionalidades da ferramenta desenvolvida e as decisões tomadas durante a sua construção; o **Capítulo 4** apresenta as considerações finais sobre o desenvolvimento do projeto e possíveis trabalhos futuros.

Capítulo 2

O SAESD

Segundo (Lachi, 2012), o Sistema de apoio à Avaliação de cursos do Ensino Superior a Distância (SAESD) é um sistema completo que integra as funcionalidades de coleta, administração e visualização dos resultados da avaliação de um curso à distância. E uma de suas principais características é a capacidade de mineração de dados computacionais capazes de auxiliar nessa tarefa de avaliação.

Uma das informações que o SAESD consegue obter são os períodos e as causas de indisponibilidade do sistema operacional no qual o curso à distância está hospedado. Para sua coleta foi desenvolvida a ferramenta de varredura de *logs* denominada SVarLog (S como referencia ao SAESD e VarLog de varredura de *log*), foi criada com a finalidade de busca e exibição *logs* de usuários que fizeram uso do sistema em um determinado período, a fim identificar os períodos de instabilidade de SOs baseados no Unix.

A ferramenta tem principal característica o uso do comando *"Last*[2]*"* e no uso do protocolo SSH para executar este comando de forma segura remotamente, sendo que na maior parte dos casos os sistemas de ensino a distancia são hospedados em sistemas Operacionais baseados no Unix.

Para que o SVarLog possa buscar os *logs* no SO servidor, a ferramenta precisa se conectar remotamente pois na maioria das vezes usuário não poderá estar fisicamente na frente do computador onde está hospedado o sistema de entrega. Para contornar esse problema, a ferramenta utiliza o protocolo de acesso remoto *Secure Shell* (SSH), seu uso se deve ao fato do SSH ser seguro e que toda a conexão é criptografada do inicio ao fim, por ele ser gratuito e vir por padrão em diversas distribuições do SO *Linux*, além de ter soluções para outras plataformas como o SO Windows. (**Ver Apêndice A,** onde é feita uma breve comparação dos protocolos Telnet e SSH**)**. Enfim, por esses motivos o

[2] Este comando exibe todas as informações referentes à entrada (*login*) e saída (*logout*) de usuários do sistema.

principal comando do SVarLog baseia-se no uso do protocolo SSH, e de combinações de comandos, "*Last"*, expressões regulares, e filtros de eventos como, *reboot*, *shutdown*, *crash*, para gerar uma saída mais amigável para o usuário.

Segundo (Lachi, 2012), a inclusão desta ferramenta no sistema SAESD foi feita através da análise da questão onde é verificado se em todos os momentos que o usuário tentou acessar o sistema de entrega, este esteve sempre disponível, se a reposta for avaliada como inadequada então é exibido um *link* de acesso para a ferramenta SVarLog, como mostrado da **Figura 2.1**. Já na **Figura 2.2** há um detalhamento e explicação das características principais da ferramenta.

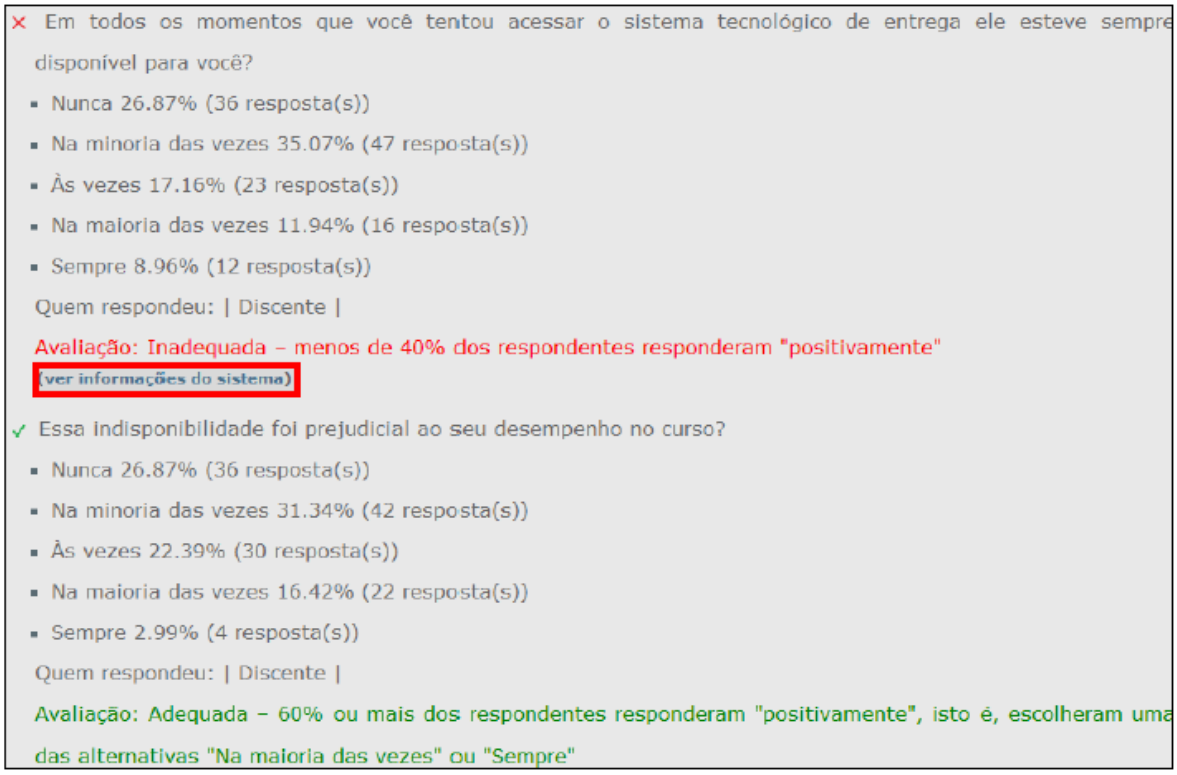


Figura 2.1. Inclusão do link para a ferramenta SVarLog. Fonte: Lachi, 2012, p. 140.

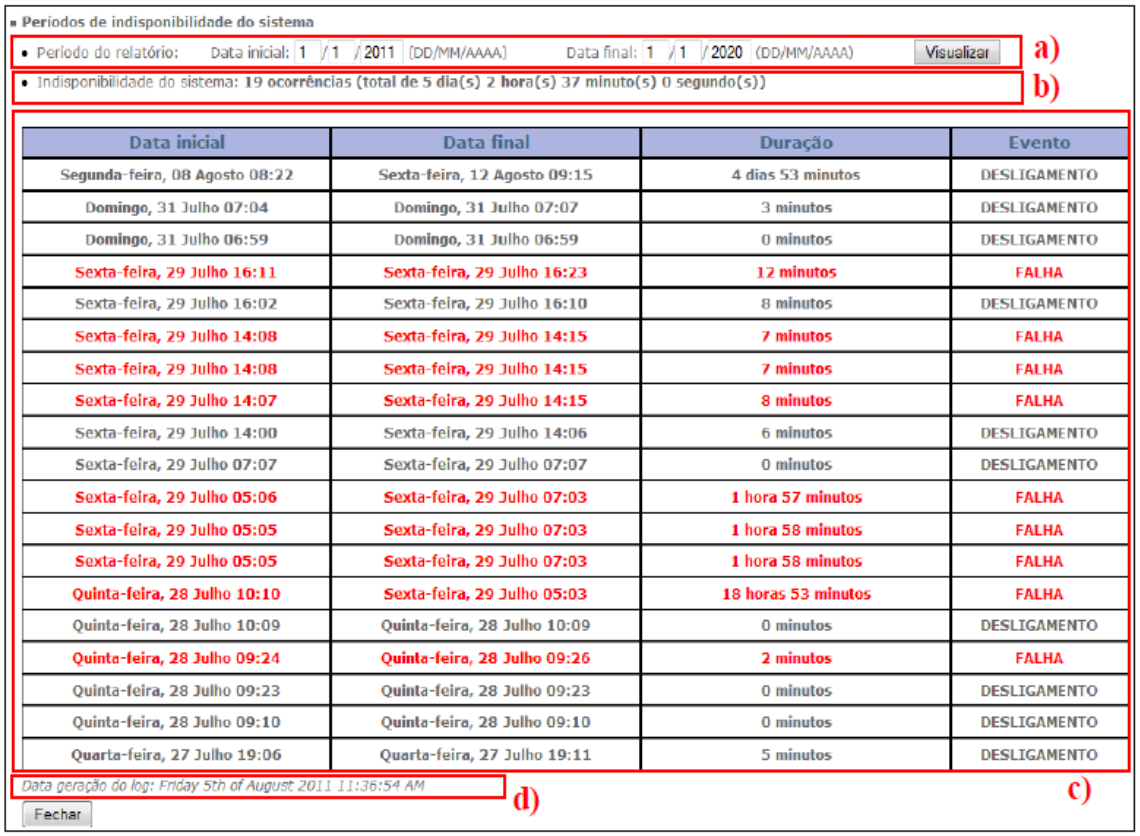


Figura 2.2. Tela do SVarLog com a exibição de todos os períodos em que o ambiente ficou indisponível. Fonte: Lachi, 2012, p. 140.

Na **Figura 2.2**, notam-se os seguintes elementos: a) dois campos que permitem especificar o período exato das ocorrências de indisponibilidade que se deseja observar; b) o número de ocorrências e a soma total de todos os períodos em que o SO ficou indisponível; c) uma tabela com a data inicial data final e duração de cada um dos momentos em que o SO ficou indisponível, com a indicação do evento que causou essa indisponibilidade – decorrência de uma falha, ou de um desligamento do sistema; d) a informação do momento em que as informações apresentadas foram extraídas do SO.

Segundo (Lachi, 2012), a interface do SVarLog foi construída para ser relativamente simples, contudo sua construção envolveu uma série de etapas e pesquisas. O primeiro ponto que deve destaque é o principal comando que a ferramenta é baseada, o comando é:

```
"ssh2 ".$user_servidor."@".$servidor." \"last -xa | grep E 'reboot|shutdown|crash'\" >
```

```
".$local.$nome_arquivo_log;
```

O comando em questão é executado através de uma chamada por linha de comando PHP, já que o sistema SAESD é feito em uma plataforma Web. Para ser possível a execução deste comando através da linguagem de programação PHP num ambiente Web e sem a necessidade de digitação de senha a cada requisição da ferramenta para buscar informações de indisponibilidade no SO Linux, foi necessário que os computadores se autenticassem de maneira a não exigir a senha de usuário ao realizar *login*. Na **Tabela 2.1** são listados todos os possíveis cenários de conexões entre as diferentes versões do protocolo SSH que a ferramenta pode ter que lidar.

	CLIENTE	**SERVIDOR**
1	**SSH-1**	**SSH-1**
2	**SSH-1**	**SSH-2**
3	**SSH-1**	**OPENSSH**
4	**SSH2**	**SSH2**
5	**SSH2**	**SSH1**
6	**SSH2**	**OPENSSH**
7	**OPENSSH**	**SSH1**
8	**OPENSSH**	**SSH2**
9	**OPENSSH**	**OPENSSH**

Tabela 2.1. Cenários possíveis em uma conexão através do protocolo SSH.

É neste contexto, que a ferramenta AutoSSH se encaixa automatizando justamente esse processo de autenticação de computadores através do conceito de gerenciamento de chaves assimétricas. Na próxima seção são mostrados, de forma geral sem expor as diferenças entre os protocolos, os passos necessários para o processo de autenticação para eliminar o uso de senhas.

2.1. *Login* automático

O esquema apresentado a seguir, ilustra passo a passo, de maneira geral, a configuração do protocolo SSH para obtenção do *login* automático sem senha. Embora os passos necessários para a configuração pareçam simples, devemos levar em conta que em todos os cenários há pequenas diferenças de um para o outro, e que sem as quais não seria possível realizar a automação da conexão. O processo para geração e gerenciamento de chaves assimétricas pública e privada devem seguir os seguintes passos:

1. Gerar as chaves assimétricas pública e privada, as chaves deverão ser geradas no cliente, como mostrado na sequência:

```
ssh-keygen
```

2. Enviar a chave pública para o servidor, a chave privada devera ficar em segurança no cliente. A chave pública será enviada para o servidor para posteriormente ser adicionada ao diretório de chaves conhecidas, isso pode ser feito usando o *Secure copy* (SCP), o comando é apresentado a seguir:

```
scp <chave pública>.pub <usuário>@<endereço remoto>: <diretório remoto do usuário>
```

3. Criar um o arquivo com o nome "*identification*" no cliente, o seu conteúdo deve conter a seguinte instrução:

```
#IdKey <chave privada>
```

4. Este quarto passo pode não ser necessário caso as versões do protocolo sejam iguais no cliente e no servidor, pois neste caso não há necessidade de conversão de chaves sendo que são compatíveis. Já para os demais cenários o tipo de formato de chaves diferem de uma versão do protocolo para outro, sendo necessária a conversão da chave, isto deverá ser feito no servidor, como mostrado a seguir:

ssh-keygen (-i ou –e) -f ~ / .ssh/<chave_cliente>.pub \ ~ / .ssh/<chave_cliente_convertida>.pub

5. Se houver conversão da chave ou não, ela deverá ser copiada para o arquivo que o servidor SSH usa para reconhecer as chaves autenticadas, esse arquivo é tem o nome de “autorized_keys” que normalmente fica no diretório “~ / .ssh / authorized_keys” no servidor SSH no SO Linux, isto é feito com o seguinte comando:

cat ~/.ssh//<chave_cliente ou chave_cliente_convertida >.~/.ssh/authorized_keys

Vale a pena destacar que para obter a autenticação sem precisar de senhas devemos ao criar as chaves, não informarmos a senha ou palavra passe para a chave, pois se for informado o usuário tem de digita-la todas as vezes que for iniciar uma conexão com o servidor, ao invés disso somente faremos a autenticação por meio das chaves assimétricas. Segundo Barrett et. al. (2005) existe outra opção, que é usar o agente SSH, o agente quando chamado é responsável por guardar a chave em memória, e usa-la em uma sessão SSH para autenticação toda vez que executar um comando, o problema no uso desta opção é que quando o usuário sair da máquina local ou encerrar o agente a configuração é perdida. Por esse motivo neste trabalho foi adotado como padrão o uso das chaves sem precisar informar a senha ou palavra passe.

Assim finalizam-se as etapas de configuração. Para verificar se a conexão não exigirá que digite uma senha, poderá ser utilizado o comando: “ssh <usuário>@<ip>” ou “ssh –l <usuário> <ip>”. O uso deste comando abrirá uma conexão com o servidor e a autenticação será através das chaves pública e privada geradas, não sendo necessário o uso de senhas. No próximo capítulo é apresentada as funcionalidades e decisões tomadas para a construção da ferramenta AutoSSH.

Capítulo 3

Ferramenta AutoSSH: automação do processo de *login* automático no SSH

O objetivo deste trabalho foi o desenvolvimento de uma ferramenta, que permite a automação do processo de autenticação por chaves assimétricas para eliminar a necessidades de digitação de senhas todas as vezes que precisasse executar um comando através de uma conexão SSH ativa.

A ferramenta em questão recebeu o nome de AutoSSH, e tem como princípio dar apoio a ferramenta de varredura de *logs* SVarLog do sistema SAESD, para eliminar a necessidade de realizar o processo de autenticação manualmente.

Embora o processo de autenticação por chaves assimétricas possa ser realizado manualmente, tornasse algo trabalhoso quando levamos em conta todos os cenários possíveis em uma conexão SSH, onde sempre haverá um cenário onde uma versão do protocolo irá interfacear com outra, por exemplo, no cliente pode estar sendo executada uma versão do protocolo SSH-2 e no servidor a versão OPENSSH.

Para que então possamos obter o efeito desejado, ou seja, *login* automático devemos, gerar as chaves assimétricas no cliente, criar arquivo que identificar a chave que foi gerada, enviar a chave do cliente para o servidor, se necessário converter a chave recebida no servidor, inserir a chave no diretório que o servidor usa para identificar os *hosts* conhecidos e por fim dar as permissões necessárias para a chave e o diretório. Devemos observar que uma versão do protocolo pode não ser compatível com a outra sendo necessárias ações diferentes para cada tipo de cenário.

A ferramenta proposta permite ao usuário obter a automação da conexão para eliminar o uso de senhas, para todos os cenários mostrados na **Tabela 2.1**, além servir de como uma interface entre o cliente e o servidor podendo executar qualquer comando operável no servidor com SO Linux, por exemplo, “ls”, para listagem de arquivos, através de uma sessão de conexão SSH.

3.1 Decisões tomadas na criação da ferramenta AutoSSH

AutoSSH foi totalmente construído na linguagem de programação C, pelo motivo desta linguagem ser portável para qualquer plataforma e também por ser uma das linguagens de programação mais utilizadas no mundo ao lado do Java, tendo destaque no mundo da programação de acordo com o índice de indicador de popularidade TIOBE *Programming Community*[3]. O índice pode ser utilizado para tomar uma decisão estratégica sobre qual a linguagem de programação que deve ser adotada na construção de um novo sistema de software. Além disso, pela razão de que ferramenta proposta será executada sob a plataforma do SO *Windows* a escolha desta linguagem esta intimamente ligada a fato de que toda a *Aplication Programmer Interface* (API) Win32[4] ser construída em C, isto só reforça sua importância.

As funções e estruturas da API Win32 foram utilizadas neste trabalho principalmente para substituir os programas auxiliares do SO Windows como mostrado na **Tabela 3.2**, para que a ferramenta fosse robusta e autônoma.

Comando/Programa	**Substituto API Win32**	**Função principal**
System(“comando”)	CreateProcess	Criar processos de execução de comandos
%username%	GetEnvironmentVariable	Retornar dados de uma determinada variável de ambiente
Mkdir	CreateDirectory	Cria um determinado diretório
Del	RemoveDirectory	Remove um determinado diretório

[3] O índice TIOBE é um indicador da popularidade de linguagens de programação e é atualizado uma vez por mês.
[4] A API Win32 é disponibilizada oficialmente pela MSDN Microsoft.

Regedit	RegCreateKeyEx	Cria chave de registro Windows
Regedit	RegSetValue	Adiciona um valor a uma chave específica
Regedit	RegOpenKey	Abre uma chave específica
Regedit	RegQueryValueEx	Lê valores de uma chave
Regedit	RegCloseKey	Fecha o registro
echo	MessageBox	Algumas saídas mais importantes foram exibidas por meio desta função da API.

Tabela 3.2. Lista de comandos/programas auxiliares do SO que foram substituídos por funções da API Win32.

3.2 Funcionamento da ferramenta AutoSSH

A ferramenta AutoSSH realiza o processo de configuração para obter a autenticação automática de acordo com os parâmetros passados durante a chamada da ferramenta. Os argumentos necessários para que a ferramenta possa funcionar são mostrados na **Tabela 3.3**.

	Parâmetro	Exemplo
a) Param1	Endereço Servidor	IP: 189.73.232.59 ou Domínio: meudominio.uems.br
b) Param2	Usuário	Aluno, Alessandro, casa etc.
c) Param3	Comando	“last -xa | grep -E 'reboot|shutdown|crash'”, “ls –la”, ou qualquer comando operável no Servidor
d) Param4	Versão SSH cliente	ssh-1, ssh1, ssh-2, ssh2, openssh
e) Param5	Versão SSH servidor	ssh-1, ssh1, ssh-2, ssh2, openssh

Tabela 3.3. Exemplo de passagem de parâmetro para a ferramenta AutoSSH.

Na **Tabela 3.3**, notam-se os seguintes parâmetros: a) o endereço do servidor ao qual deseja iniciar uma conexão; b) o nome do usuário que tenha uma conta cadastrada no servidor; c) o comando que será executado na máquina remota; d) a versão do protocolo SSH do cliente; e) a versão SSH do servidor.

A chamada da ferramenta juntamente com a passagem de parâmetro fica da seguinte forma:

```
AutoSSH.exe <EndServidor> <Usuario> <Comando> <VerCliente> <VerServidor>
```

Quando os parâmetros são informados corretamente para a ferramenta, a primeira função do programa é identificar que tipo de conexão foi requisitado, sendo

que para essa identificação foram numeradas de 1 a 3 respectivamente, todas as atuais versões do protocolo SSH até a presente data deste trabalho, a versão SSH-1, SSH-2 e OPENSSH. A partir desta premissa foram criadas 9 combinações numéricas com as quais foram elaboradas as 9 definições para representar computacionalmente todos os possíveis cenários para uma conexão através do protocolo SSH, como representado na **Tabela 3.4.**

Cliente x Servidor	Combinação	#DEFINE para a linguagem C
SSH-1 x SSH-1	11	SSH1_PARA_SSH1
SSH-1 x SSH-2	12	SSH1_PARA_SSH2
SSH-1 x OPENSSH	13	SSH1_PARA_OPENSSH
SSH-2 x SSH-1	21	SSH2_PARA_SSH1
SSH-2 x SSH-2	22	SSH2_PARA_SSH2
SSH-2 x OPENSSH	23	SSH2_PARA_OPENSSH
OPENSSH x SSH-1	31	OPENSSH_PARA_SSH1
OPENSSH x SSH-2	32	OPENSSH_PARA_SSH2
OPENSSH x OPENSSH	33	OPENSSH_PARA_OPENSSH

Tabela 3.4. Esquema para identificação dos diferentes cenários possíveis em uma conexão SSH.

Depois de identificado o tipo de cenário que a ferramenta irá trabalhar, o AutoSSH faz a leitura do registro do SO Windows, através de funções da API Win32 exemplificadas na **Tabela 3.2**, dados de uso anterior da ferramenta, caso o sistema identifique que o endereço de servidor, usuário, versão do cliente SSH e versão do servidor SSH, são diferentes dos armazenados em registro, o sistema passa a executar a função que realiza toda configuração e gerenciamento das chaves assimétricas para não ser necessário o uso de senhas.

Se após verificação for constatado que são os mesmo dados, o sistema identifica que já foi realizado o processo de automação, então é aberta uma conexão com o servidor usando as configurações já realizadas para executar o comando passado por parâmetro, útil caso o usuário da ferramenta realize muitas iterações com o servidor.

3.3 Principal função do AutoSSH

A principal função da ferramenta proposta foi denominada “gera_autenticacao”, justamente pela sua utilidade, esta é a função mais importante do trabalho, pois é ela que direciona e gerencia toda a autenticação de chaves assimétricas. Essa função recebe como parâmetro uma estrutura AUTO_SSH e o identificador de cenário como um tipo de dado numérico, por exemplo, 22 que representa a definição SSH2_PARA_SSH2 de acordo com a **Tabela 3.4.**

A estrutura AUTO_SSH foi montada para representar no programa as propriedades básicas que o AutoSSH precisa para execução, como mostrado na **Tabela 3.5**.

```
typedef struct {
         1) char Servidor[TAMANHO];
         2) char Usuario[TAMANHO];
         3) char Comando[TAMANHO];
         4) char VerCliente[TAMANHO];
         5) char VerServidor[TAMANHO];
         6) char LigacaoSimbolica[TAMANHO];
} AUTO_SSH;
```

Tabela 3.5. Representação da estrutura AUTO_SSH da ferramenta AutoSSH.

Na **Tabela 3.5** podemos notar as seguintes propriedades: 1) Servidor, variável que armazena o endereço do servidor; 2) Usuário, para armazenar o nome de usuário; 3) Comando, armazena o comando passado para execução no servidor; 4) VerCliente, armazena a versão SSH do cliente; 5) VerServidor, armazena a versão SSH do servidor; 6) LigacaoSimbolica, variável usada para representar os prefixos de comando de cada versão do SSH, por exemplo a versão SSH-2 tem ligação simbólica “ssh2”, enquanto a versão OpenSSH tem ligação simbólica “ssh”.

Como as definições representam os cenários e os identificadores os numerais de combinação, que ao serem comparados dão exatamente os tipos de cenários que a ferramenta precisa trabalhar, esta função ficou responsável por direcionar o fluxo de execução do programa para cada bloco de implementação diferenciada para cada

cenário, sendo que assim se faz necessário, pois cada cenário tem a sua peculiaridade e distinção dos demais, alguns são praticamente iguais, mas para manter a lógica de programação foi colocado cada um em um bloco.

As principais diferenças de configuração nos diferentes cenários estão na conversão ou não da chave no servidor, que cada versão gera. Quando o cenário de conexão é de duas versões iguais não é necessário a conversão das chaves pois são compatíveis, já quando as versões são diferentes há necessidade de conversão da chave. Ainda deve-se levar em conta que até mesmo para a conversão há diferenças, como mostrado na **Tabela 3.6**, que expõe as diferenças nos comandos utilizados para tais fins.

Cenário	Comando de conversão de chave
SSH1_PARA_OPENSSH	ssh-keygen -i -f <chave_publica>
SSH1_PARA_SSH2	ssh-keygen -e -f <chave_publica>
SSH2_PARA_SSH1	ssh-keygen2 -e -f <chave_publica>
SSH2_PARA_OPENSSH	ssh-keygen2 -i -f <chave_publica>
OPENSSH_PARA_SSH1	ssh-keygen -e -f <chave_publica>
OPENSSH_PARA_SSH2	ssh-keygen -e -f <chave_publica>

Tabela 3.6. Exemplo de diferenças de conversão de chaves nos diferentes cenários.

A ferramenta AutoSSH além ter sua construção justificada pelo auxílio ao SVarLog, também pode ser utilizada avulso servindo como um software facilitador na administração de um servidor baseado no Unix, pois é possível executar quaisquer comando operável em um servidor baseado no Unix através do AutoSSH.

Para ver como foi montado o ambiente de teste consulte o **Apêndice B** e consulte o **Apêndice C**, onde é apresentada a execução da ferramenta no cenário SSH2_PARA_OPENSSH.

Capítulo 4

Conclusão

Neste trabalho foi o apresentado o desenvolvimento de uma ferramenta capaz de realizar autonomamente uma conexão remota por meio do protocolo SSH, de modo a subsidiar a coleta de dados do SAESD.

Para que a ferramenta fosse o mais independente possível de comandos existentes no SO, suas principais funções foram desenvolvidas através do uso de estruturas e funções da API Win32 disponibilizadas pela MSDN Microsoft. Como essa API foi totalmente construída em C, mesma linguagem em que a ferramenta foi construída, a utilização dessa API foi feita diretamente no código por meio de chamadas à API Win32.

Os testes foram realizados em uma conexão hipotética entre um SO Windows com o programa SSH cliente e uma máquina virtual instalada no próprio SO Windows onde foi instalado o SO OpenSUSE para simular o servidor contendo o programa SSH servidor. O processo de conexão foi realizado com sucesso para os cenários SSH2_PARA_OPENSSH e OPENSSH_PARA_SSH2. Para os cenários envolvendo a versão SSH-1, apenas o cenário SSH1_PARA_OPENSSH foi considerado aceitável. No cenário SSH1_PARA_SSH2 houve problemas resultantes de incompatibilidades no formato das chaves.

4.1 Trabalhos futuros

Como possibilidades de trabalhos futuros podem ser listados:

- Podem ser programados mecanismos de identificação prévia das versões do protocolo SSH envolvida na conexão, tanto no cliente como no servidor, para maior automação da ferramenta, não precisando ser passados como parâmetro;
- Realizar a portabilidade desta ferramenta para outros SOs, afim de evitar a restrição deste *software* apenas para o SO Windows, mas para outros sistemas operacionais, como as muitas distribuições do SO Linux.

Referências Bibliográficas

BARRETT, Daniel J. SILVERMAN Richard E. e BYRNES Robert G. **O'REILLY "SSH: The Secure Shell - The Definitive Guide".** 2ª Edição, 2005.

LACHI, Ricardo L. **SAESD - Sistema de apoio à Avaliação de cursos do Ensino Superior a Distância (SAESD)**. Data da defesa: 27 de novembro 2012. 347 f. Tese (Doutorado em Ciências da Computação) – Universidade Estadual de Campinas, Campinas, 2012.

Índice TIOBE *Programing Community*

Disponível em: <http://www.tiobe.com/tiobe-index/>. Acessado em 22/09/16.

MORIMOTO, Carlos E. **Redes e Servidores Linux**. GDH Press e Sul Editores 2ed, 2006.Disponível em <http://www.hardware.com.br/livros/linux-redes/usando-ssh.html>. Acessado em 15/03/2016.

MSDN – Microsoft. The Windows application programming interface (API) .

Disponível em: <https://msdn.microsoft.com/en-us/library/windows/desktop/hh920508(v=vs.85).aspx>. Acessado em 21/08/16.

RAMESH, Natarajan **SSH2 PARA SSH2**

Disponível em: <http://www.thegeekstuff.com/2008/06/comprehensive-guide-for-ssh2-key-based-authentication-setup/>. Acessado em 11/07/2016

RAMESH, Natarajan **OPENSSH PARA SSH2**

Disponível em: <http://www.thegeekstuff.com/2008/07/ssh-key-based-authentication-setup-from-openssh-to-ssh2/>. Acessado em 11/07/2016.

RAMESH, Natarajan **SSH2 PARA OPENSSH**

Disponível em: <http://www.thegeekstuff.com/2008/10/perform-ssh-and-scp-without-password-from-ssh2-to-openssh/>. Acessado em 11/07/2016.

RAMESH, Natarajan **OPENSSH PARA OPENSSH**

Disponível em: <http://www.thegeekstuff.com/2008/06/perform-ssh-and-scp-without-entering-password-on-openssh/>. Acessado em 11/07/2016.

TANENBAUM, Andrew S. **Redes de Computadores**. Rio de Janeiro: Elsevier. Trad.Vanderberg de Souza. 4ª Edição, 2003. 632 Páginas.

YLONEN, Tatu. **RFC 4251.SSH Communications Security Corp.** Cisco Systems, Inc. 2006.Disponível em: <http://tools.ietf.org/html/rfc4251>. Acessado em 05/04/2016.

Apêndice A

Protocolo Telnet x protocolo SSH

Protocolo é um conjunto de regras e procedimentos que permitem a comunicação entre processos, isto é, estabelece regras a serem respeitadas para envio e recebimento de informações.

Segundo Morimoto (2006), o Telnet é um protocolo primitivo que permite rodar comandos remotamente através de uma interface de modo texto. Existem clientes Telnet para vários sistemas operacionais. Tanto no SO Linux quanto no SO Windows, para acessar uma máquina remotamente via Telnet usa-se o comando "telnet" seguido do endereço IP destino. O grande problema é que o Telnet não oferece nenhum tipo de segurança. Todas as informações, incluindo as senhas são transmitidas em texto puro, de forma legível pela rede e são fáceis de interceptar. A F**igura A.1** apresenta uma conexão remota através do Telnet.

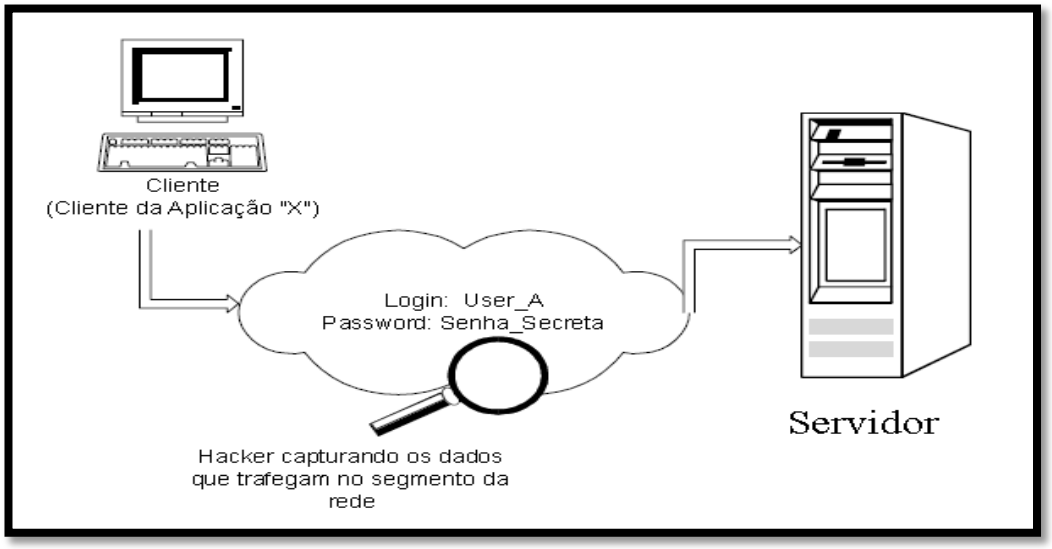


Figura A.1: Comunicação remota com o protocolo Telnet.

Segundo Barrett et. al. (2005) o protocolo SSH foi desenvolvido em 1995 por Tatu Ylonen, fundador da empresa *SSH Communications Security*, teve sua primeira versão denominada SSH1, que se manteve como software livre até a versão seguinte SSH2, que é atualmente a versão mais utilizada. O Secure Shell (SSH) é um protocolo

para *login* remoto seguro e outros serviços de segurança de rede em uma rede insegura. A grande vantagem do SSH, sobre outros protocolos é a ênfase em segurança. O SSH foi desenvolvido principalmente com a intenção de substituir os protocolos inseguros que permitem que o usuário se conecte remotamente a outro computador de forma segura. Um servidor SSH bem configurado é virtualmente impenetrável podendo ser acessado de forma segura, mesmo que a rede esteja comprometida.

Segundo Morimoto (2006),o SSH é usado em tantos servidores importantes que uma brecha grave poderia (literalmente) parar o mundo. Por isso, todo o código é exaustivamente auditado por uma variedade de empresas e órgãos governamentais. O SSH utiliza um conjunto de técnicas de criptografia para assegurar que somente apenas as pessoas autorizadas terão acesso ao servidor; que todos os dados transmitidos por meio do uso deste protocolo, sejam impossíveis de decifrar; que a integridade da conexão seja mantida.São previstas respostas para diversos tipos de ataques conhecidos. O SSH detecta casos em que o servidor tenha sido substituído por outra máquina, situações nas quais se tenta injetar dados na conexão, ou seja, tirar proveito de uma conexão aberta para incluir pacotes com comandos adicionais e incluir até mesmo técnicas de "despiste", que tornam muito mais complicado descobrir em qual pacote cifrado foi transmitida a senha de acesso, por exemplo, dificultando a descoberta da senha usando um ataque de força-bruta[5].A idéia central é que, mesmo em situações onde seja fácil interceptar a transmissão, como no caso de uma rede wireless pública, seja impossível descobrir o conteúdo dos pacotes, devido à criptografia.

Um dos algoritmos de criptografia de dados que o SSH usa é o RSA. De acordo com Tanenbaum (2003), O RSA foi inventado em 1978 por Ron L. Rivest, AdiShamir e LenAdleman que, na época, trabalhavam no *MassachussetsInstituteof Technology*(M.I.T.). As letras RSA correspondem às iniciais dos inventores do código. Esse algoritmo de chave pública tem a sua força na primitiva de que é muito difícil a fatoração de números grandes. Por exemplo, se tivéssemos uma chave de 500 dígitos, com um computador de milhares de instrução por segundo, levaria cerca de 10^{25} anos para decifrar a chave e assim poder decifrar uma mensagem codificada. Mesmo que os computadores evoluíssem gradativamente com o passar do tempo, nossos sucessores no futuro poderiam simplesmente escolher dígitos cada vez maiores, isso o torna

[5]Um ataque de força bruta, ou *brute force*, consiste em adivinhar, por tentativa e erro, um nome de usuário e senha e, assim, executar processos e acessar sites, computadores e serviços em nome e com os mesmos privilégios deste usuário.

indecifrável. Felizmente, os matemáticos têm tentado fatorarnúmeros extensos por pelo menos 300 anos, e o conhecimento acumulado sugere que o problema é extremamente difícil.

O protocolo SSH permite criar túneis criptografados entre dois computadores. Uma vez criado o túnel, é possível transmitir qualquer tipo de dados de forma segura, também é possível usar ferramentas conjunta com o SSH, por exemplo, SCP e SFTP para transferência de arquivos entre as máquinas, uma vez que tudo que passa pela rede é criptografado pelo SSH. A **Figura A.2** mostra a conexão remota com o uso do protocolo SSH.

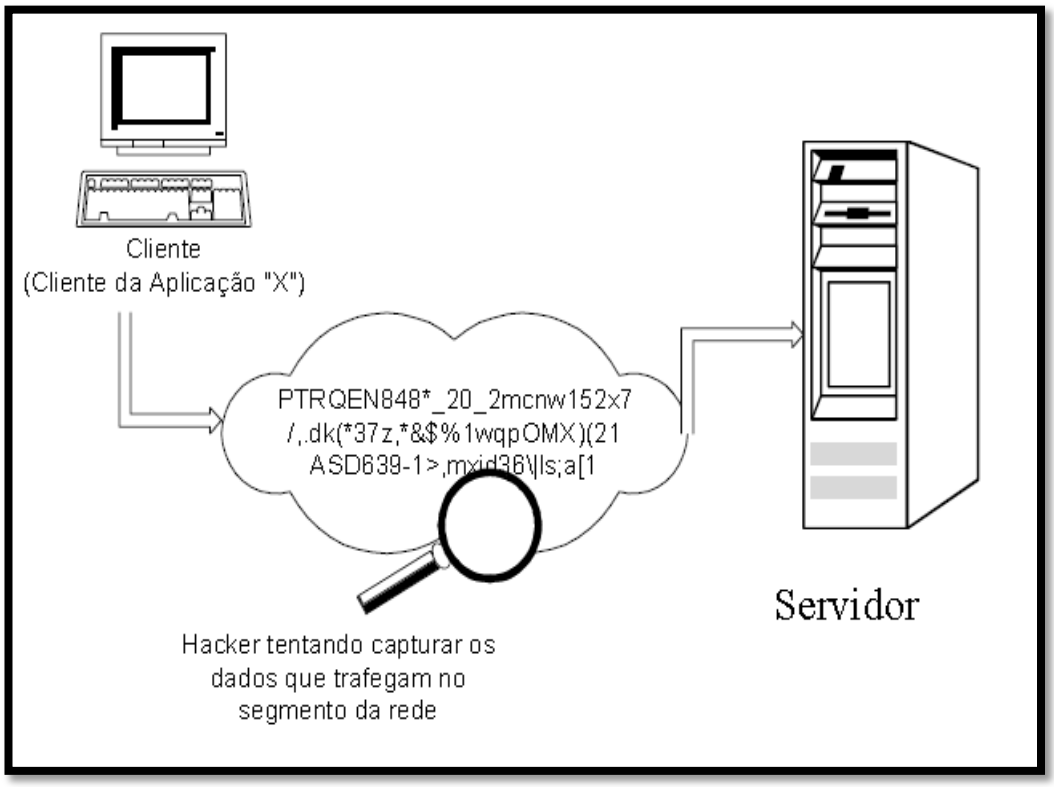


Figura A.2: Conexão remota com o protocolo SSH.

Apêndice B

Montagem do ambiente de testes

Protocolo SSH no SO Linux

Para o exemplo de software SSH para o SO Linux, foi escolhido o *OpenSSH*, uma versão do protocolo SSH de licença livre que vem na maioria das distribuições do SO Linux atualmente. Nesse exemplo será mostrado como iniciar o serviço SSH no SO Linux e como configurá-lo para aceitar receber conexões SSH na porta 22, porta padrão para conexões SSH, com o objetivo de montar um servidor para o cliente SSH do SO Windows. No primeiro exemplo inicia-se o serviço SSH servidor, na máquina OpenSuse:

Exemplo 1, inicialização do servidor SSH no OpenSuse:

```
sudo /etc/init.d/sshd start
```

No segundo exemplo é mostrado como configura o *firewall* do SO Linux para aceitar conexões SSH na porta 22, mas fica a critério do usuário escolher qualquer porta que achar segura, uma vez que por padrão o *firewall* bloqueia portas de acesso a máquina:

Exemplo 2, configuração do *firewall* para aceitar conexão na porta 22:

```
iptables -I INPUT -p tcp --dport 22 -j ACCEPT
```

Com o servidor devidamente configurado e em posse de seu endereço IP[6], a próxima etapa será a preparação da máquina com SO Windows com o cliente SSH instalado.

[6]IP - *Internet protocol*: é um protocolo de comunicação usado entre duas ou mais máquinas em rede para encaminhamento de dados.

Protocolo SSH no SO Windows

Um exemplo de programa SSH para o SO Windows é o *SSH communications security*, *software* de licença comercial, para universidades e uso doméstico. Para exemplo foi utilizada a versão SSH2 do protocolo, por se tratar da versão mais utilizada. A utilização do cliente SSH para conexão remota é feita da seguinte forma:

Exemplo 1:

ssh2 -l <usuário> <ip> ou ssh2 -l <ip>

Exemplo 2:

ssh2 <usuário>@<ip>

De qualquer forma é necessário que seja informado o usuário que esteja devidamente cadastrado na lista de usuários do servidor, através de uma conta protegida por senha. Também é necessário informar o endereço IP do servidor. Na primeira forma é usado o comando "-l" para especificar o usuário e o IP, mas é possível ocultar o usuário. Nesse caso, o SSH tentará fazer *login* com o mesmo nome de usuário da máquina cliente que está requerendo a conexão no servidor. Na segunda forma o comando "@" é utilizado sendo obrigatórios a informação do usuário e o IP para que o SSH possa requerer a conexão com o servidor SSH.

Como o servidor já esta aceitando conexões SSH, ao ser solicitado uma conexão ele precisa conhecer a autenticidade do requerente, no caso o cliente SSH do SO Windows. Para isso o servidor SSH pede a senha da conta do usuário que foi informado, se a resposta, ou seja, a senha, for compatível com a cadastrada no servidor, o SSH libera a conexão com o servidor, no nosso exemplo a máquina com o OpenSuse.

Apêndice C

Execução do AutoSSH: cenário SSH2_PARA_OPENSSH

Neste apêndice é mostrado a execução da ferramenta AutoSSH para o cenário onde a versão SSH-2 é o cliente no SO Windows e a versão OPENSSH é o servidor no OpenSUSE.

a) Chamada da ferramenta e passagens de parâmetro para a mesma;

```
C:\Users\Alessandro\Desktop>AutoSSH.exe 192.168.3.69 alessandro "last -xa | grep
 -E 'reboot|shutdown|crash'" ssh2 openssh
```

b) Identificador de cenário de conexão SSH.

```
Identificador: 23.
```

c) Geração de chaves assimétricas;

```
Gerando chaves.....
Atencao, nao digite nada logo apos passphrase e em again. So pressione enter.
Se quizer voce pode criar uma senha, digite-a apos passphrase,repita em again.
OBS: Se criar uma senha tera que digita-la toda vez que acessar o linux.
Generating 1024-bit dsa key pair
   1 oOo.oOo.oOo.
Key generated.
1024-bit dsa, Alessandro@ALESSANDRO-PC, Wed Sep 28 2016 16:13:25
Passphrase :
Again      :
Key is stored with NULL passphrase.
 (You can ignore the following warning if you are generating hostkeys.)
 This is not recommended.
 Don't do this unless you know what you're doing.
 If file system protections fail (someone can access the keyfile),
 or if the super-user is malicious, your key can be used without
 the deciphering effort.
Private key saved to C:/Users/Alessandro/Application Data/SSH/UserKeys/id_dsa_10
24_a
Public key saved to C:/Users/Alessandro/Application Data/SSH/UserKeys/id_dsa_102
4_a.pub
```

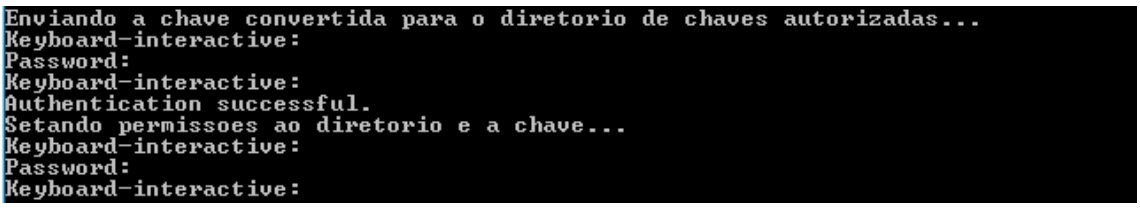
d) Envio da chave para o servidor;

```
Preparando para enviar chave para servidor...
Digite sua senha de usuario linux:
Password:
id_dsa_1024_a.pub                         |  749B |  749B/s | TOC: 00:00:01 | 100%
```

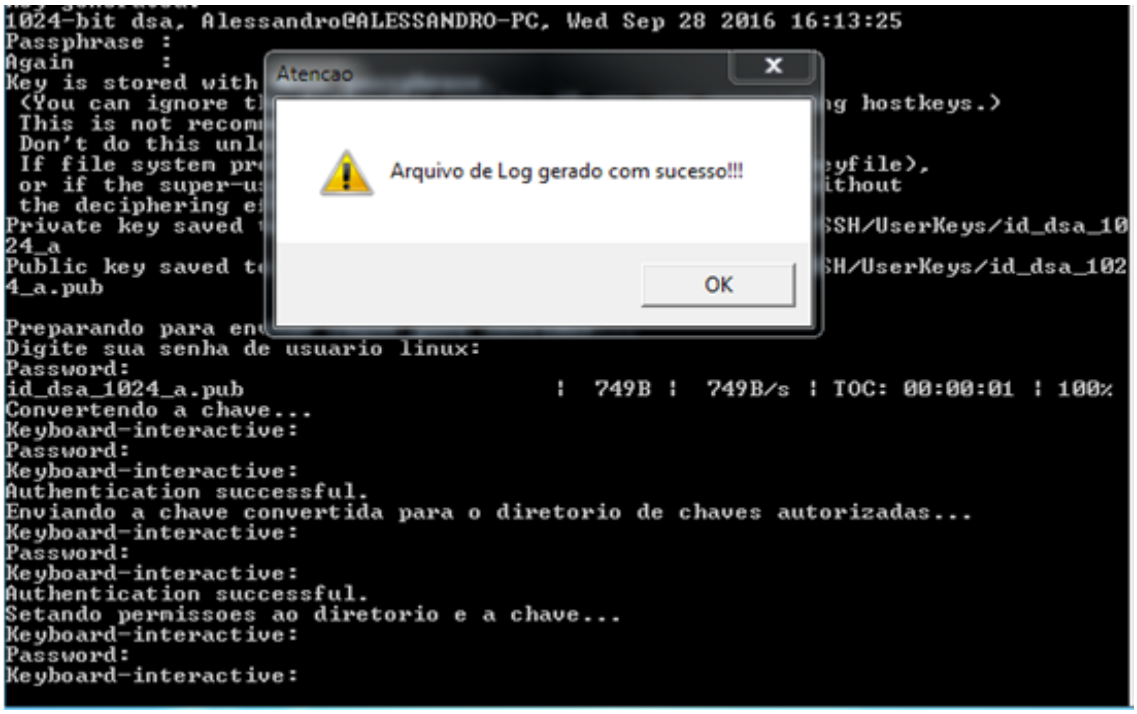
e) Conversão da chave no servidor;

```
Convertendo a chave...
Keyboard-interactive:
Password:
Keyboard-interactive:
Authentication successful.
```

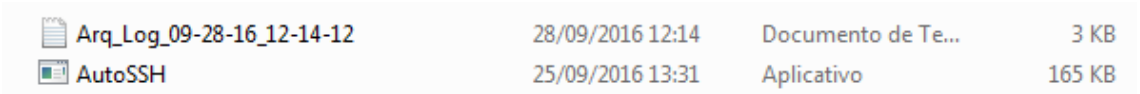
f) Envio da chave convertida para o diretório de reconhecimento de chaves autorizadas e “setando” as permissões de acesso à chave e ao diretório;



g) A saída é um arquivo de *log*, no formato (mm-dd-yy_hh-mm-ss), com o resultado do comando executado no servidor;



h) O arquivo de *log* é gerado no mesmo diretório da ferramenta;



i) Conteúdo do arquivo de *log*.

```
reboot   system boot  Wed Sep 28 11:57 - 12:14  (00:16)     2.6.34-12-default
alessand pts/2        Mon Sep 26 17:20 - crash (1+18:36)
alessand pts/1        Mon Sep 26 17:20 - crash (1+18:36)
alessand :0           Mon Sep 26 17:20 - crash (1+18:37)    console
reboot   system boot  Mon Sep 26 17:19 - 12:14 (1+18:54)    2.6.34-12-default
alessand pts/3        Mon Sep 19 17:15 - crash (7+00:04)
alessand pts/2        Mon Sep 19 17:15 - crash (7+00:04)
alessand :0           Mon Sep 19 17:14 - crash (7+00:04)    console
reboot   system boot  Mon Sep 19 17:14 - 12:14 (8+18:59)    2.6.34-12-default
alessand pts/3        Mon Sep 19 12:43 - crash  (04:30)
alessand pts/2        Mon Sep 19 12:43 - crash  (04:30)
alessand :0           Mon Sep 19 12:43 - crash  (04:31)     console
reboot   system boot  Mon Sep 19 12:42 - 12:14 (8+23:31)    2.6.34-12-default
alessand pts/0        Tue Sep 22 00:46 - crash (363+11:56)
alessand pts/2        Mon Sep 21 18:17 - crash (363+18:24)
alessand :0           Mon Sep 21 18:17 - crash (363+18:25)  console
reboot   system boot  Mon Sep 21 18:17 - 12:14 (372+17:57)  2.6.34-12-default
alessand pts/0        Sat Mar 21 07:47 - crash (184+10:29)
alessand pts/3        Fri Mar 20 20:21 - crash (184+21:55)
alessand pts/2        Fri Mar 20 20:21 - crash (184+21:55)
alessand :0           Fri Mar 20 20:21 - crash (184+21:55)  console
reboot   system boot  Fri Mar 20 20:21 - 12:14 (557+15:53)  2.6.34-12-default
alessand pts/0        Thu Mar 12 19:07 - crash (8+01:13)
alessand pts/2        Thu Mar 12 18:32 - crash (8+01:48)
alessand :0           Thu Mar 12 18:32 - crash (8+01:48)    console
reboot   system boot  Thu Mar 12 18:31 - 12:14 (565+17:42)  2.6.34-12-default
shutdown system down  Thu Mar 12 18:30 - 18:31  (00:01)     2.6.34-12-default
reboot   system boot  Thu Mar 12 18:16 - 18:30  (00:14)     2.6.34-12-default
shutdown system down  Tue Mar 10 19:41 - 18:16 (1+22:34)    2.6.34-12-default
reboot   system boot  Tue Mar 10 19:24 - 19:41  (00:17)     2.6.34-12-default

```