José Lopes de Oliveira Júnior

Windows Registry Fixer (WRF): Automação na Criação de *Scripts* para Controle de Permissões do Windows através do Samba

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Prof. Herlon Ayres Camargo

Lavras
Minas Gerais - Brasil
2009

José Lopes de Oliveira Júnior

Windows Registry Fixer (WRF): Automação na Criação de *Scripts* para Controle de Permissões do Windows através do Samba

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em *21 de Novembro de 2009*

Prof. Denilson Vedoveto Martins

Profa. Marluce Rodrigues Pereira

Prof. Herlon Ayres Camargo
(Orientador)

Lavras
Minas Gerais - Brasil
2009

Dedico esta monografia à comunidade do software livre, que vem trabalhando com empenho durante anos para prover software de qualidade a todos, livre de patentes ou impedimentos de qualquer empresa.

Agradecimentos

Agradeço aos meus pais que me deram apoio e incentivo para estudar.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho apresenta uma forma para criação automatizada de *scripts* de configuração para o Registro do Windows. Além disso é mostrado como utilizar estes *scripts* em conjunto com um servidor Samba, para ditar as permissões de utilização das máquinas clientes em um domínio Windows.

Palavras-Chave: Windows; Samba; Interoperabilidade; Linux; Python.

Capítulo 1

Introdução

De acordo com (Net Applications, 2009), o Microsoft Windows[1] é o sistema operacional (SO) mais utilizado em todo o mundo, com cerca de 90% de utilização.

Criticado por usuários em geral, devido à sua falta de segurança, o Windows é realidade nos parques computacionais de várias empresas, pelo seu uso em massa. Desta forma, profissionais da área de Tecnologia da Informação (TI) não podem ignorar a sua existência, ainda que muitos não sejam favoráveis quanto à utilização do mesmo. O que resta para profissionais que precisam trabalhar com este sistema é aprender a lidar com ele, tornando-o mais seguro e protegido de pragas virtuais.

É de suma importância que administradores de sistemas computacionais pensem na segurança e bom uso dos mesmos, pois isto garante uma melhor vida útil dos computadores e diminui o risco de perda ou roubo de dados da empresa.

Muitas empresas possuem sistemas Windows instalados em suas máquinas, sistema este que é alvo da maior parte das pragas virtuais existentes. Por isso o administrador deve tomar diversas precauções de forma a aprimorar a segurança deste sistema operacional. Grande parte desse trabalho pode ser realizada pela edição direta do Registro do Windows, o que, apesar de ser altamente perigoso para o sistema, permite que a tarefa seja automatizada com a utilização de *scripts* que, quando importados pelo Windows, podem configurá-lo em vários aspectos definidos pelo administrador.

[1] Por ser amplamente conhecido como "Windows", este texto utilizará esta denominação para se referir a este sistema da Microsoft.

Com o crescente avanço do Linux, estão se tornando comuns ambientes providos de servidores com este SO, que são usados como controladores de domínio para redes Windows. Ambientes deste tipo tornam-se possíveis graças ao Samba, um *software* responsável por promover a interoperabilidade entre sistemas Unix (o que inclui o Linux) e Windows.

1.1 Objetivo

Este trabalho almeja criar uma forma mais simples e intuitiva para gerar *scripts* de configuração para o Registro do Windows. Além disso busca-se aplicar uma solução onde o Linux age, através do Samba, como um controlador de domínio para clientes Windows, mas com um diferencial: em vez de se limitar a simplesmente centralizar as contas dos usuários, o controlador de domínio definirá as permissões de utilização das máquinas que entrarem no domínio, através dos *scripts* previamente gerados. Tudo isso como forma de melhorar a segurança das máquinas clientes com relação ao mau uso por parte dos usuários e para ajudar o administrador na implementação das suas Políticas de Segurança.

1.2 Justificativa

Este trabalho se justifica pela necessidade crescente em garantir e melhorar a interoperabilidade entres diferentes SO's, por apresentar uma solução que está se tornando comum atualmente (servidores Linux para clientes Windows) e por facilitar a criação de *scripts* para o Registro do Windows, uma vez que a sintaxe de tais *scripts* não é intuitiva.

Além disso, a constante preocupação com a segurança dos SO's e com formas de se implementar Políticas de Uso e Segurança, por parte de administradores de sistemas, deve tornar o presente documento bastante útil para tais profissionais.

1.3 Metodologia

Para o desenvolvimento deste trabalho foi feita, inicialmente, uma revisão sobre Políticas de Uso e Segurança, sobre o núcleo de configurações do Windows (Registro do Windows), sobre o Samba e sobre a linguagem Python (usada para confecção de um módulo para criação de *scripts* para o Registro do Windows).

Em seguida é mostrado como todas as soluções apresentadas se integram, possibilitando a realização do objetivo proposto, com todas as etapas descritas. Isto inclui o desenvolvimento de um módulo em Python para automatizar a criação de *scripts* para o registro do Windows. Finalmente são apresentados os resultados obtidos, incluindo os testes que foram feitos no ambiente criado, para verificar se os objetivos do trabalho foram alcançados, seguido pelas conclusões e propostas para trabalhos futuros.

1.4 Desenvolvimento

A estrutura deste trabalho é composta por cinco capítulos, sendo o primeiro reservado à introdução do assunto. No Capítulo 2 é feita a revisão da bibliografia, onde cada solução utilizada para realização do trabalho é apresentada. Em seguida, no Capítulo 3, é descrito como cada solução apresentada no Capítulo 2 foi usada em combinação com as demais para que o objetivo fosse atingido.

No Capítulo 4 são apresentados os resultados obtidos com o desenvolvimento deste trabalho, detalhando como foram feitos os testes e o Capítulo 5 é composto pela conclusão e pelas propostas para trabalhos futuros, que apresentam ideias para melhorar e estender o uso deste trabalho.

Capítulo 2

Revisão Bibliográfica

Este capítulo apresenta noções básicas sobre competências essenciais para o desenvolvimento do trabalho. Primeiramente são apresentadas as Políticas de Uso e Segurança. Este assunto é explorado de forma a lembrar o leitor sobre a importância da definição de uma boa política antes de se implementar qualquer tipo de regra de utilização na rede de computadores. Em seguida é apresentado o Registro do Windows com seu histórico, definições e um guia básico para trabalhar com o mesmo.

Na sequência é mostrado o servidor Samba, fundamental para integração entre redes Windows/Unix. É apresentado um breve histórico do programa, seguido por noções básicas de configuração do mesmo. Por fim, a linguagem de programação Python é apresentada com um roteiro similar aos tópicos anteriores: história resumida e noções básicas com exemplos de aplicações diretas para o trabalho.

2.1 Políticas de Uso e Segurança

De acordo com (NBSO, 2003), a política de segurança é um documento que atribui direitos e responsabilidades às pessoas que lidam com os recursos computacionais de uma instituição e com as informações neles armazenados. É um manual que serve para amparar o administrador da rede nas decisões relativas à segurança dos recursos de Tecnologia da Informação (TI). Tal política deve conter todas as tarefas que estão previstas para os usuários executarem na rede, tudo o que lhes é negado, além de sansões para aqueles que se envolverem com casos de não conformidadade com as diretrizes previstas.

Ainda segundo (NBSO, 2003), uma seção que normalmente acompanha a política de segurança é a política de uso aceitável (AUP, da sua sigla em inglês: *Acceptable Use Policy*). Esta política define como poderão ser usados os recursos de TI na organização e é recomendável, de acordo com (NBSO, 2003), que a utilização seja atrelada a uma concordância expressa dos usuários aos termos da política, como em um contrato de prestação de serviços.

Conforme (NBSO, 2003), a política de segurança é mais um mecanismo para evitar a quebra de uma ou mais das três propriedades fundamentais da segurança da informação: confidencialidade, que limita o acesso à informação às entidades legítimas; integridade, que garante que a informação mantenha os dados definidos pelo seu criador; e disponibilidade, que garante o uso legítimo da informação a qualquer momento.

(NBSO, 2003) pondera que não existe uma metodologia para criação de políticas de segurança, mas afirma que uma análise de riscos deve preceder a criação da política, de modo com que o administrador possa definir claramente qual é a informação que deverá ser protegida pela política e de onde vêm os riscos que estas informações correm. A mesma fonte ainda enumera tópicos que devem ser abordados na política: aspectos preliminares (e.g., meios de distribuição da política e revisões), política de senhas (e.g., requisitos para formação e normas para proteção de senhas), direitos e responsabilidades dos usuários (e.g., utilização das contas de acesso e uso aceitável de email e acesso à Web), direitos e responsabilidades do provedor (e.g., *backups* e normas de segurança física), sansões (e.g., penalidades cabíveis em caso de violação da política).

Como em muitos casos, o administrador da rede é subordinado a alguém dentro da empresa, (NBSO, 2003) indica que o apoio da administração superior é um dos pontos que influencia no sucesso da aplicação da política. Outros pontos seriam a amplitude da política, que deve cobrir os aspectos tangíveis aos recursos de TI da empresa; as revisões, que devem seguir as mudanças da empresa; a vigilância constante, que deverá garantir o contínuo respeito à política; o anúncio formal da política, que deverá informar todos os usuários sobre a utilização de tal política, fazendo com que os mesmos se submetam a ela antes de terem contato com os recursos de TI; e a disponibilidade da política, que deve ser de fácil acesso a qualquer um dentro da empresa.

Em contraste, (NBSO, 2003) cita causas comuns de fracasso de políticas de segurança, que são: políticas demasiadamente detalhadas ou restritivas (deve-se usar abstração para eliminar informações desnecessárias, além de saber dosar as restrições, para não prejudicar o trabalho dos usuários), não devem haver excessões

(todos os grupos que estão submetidos à política devem respeitá-la igualmente, sem protecionismos por parte do administrador) e a política não pode se ater a *softwares* ou *hardwares* específicos (em vez de detalhar estes ítens ao nível de nomes ou marcas, deve-se generalizar, pensando em futuras decisões de mudança de produtos).

2.2 Registro do Windows

De acordo com (HONEYCUTT, 2005), o Registro do Windows é um banco de dados hierárquico, em forma de árvore, que pode ser descrito como um repositório central para dados de configuração do SO e aplicativos de usuário. Através da edição direta do Registro é possível realizar configurações no sistema que não estão disponíveis através da interface com o usuário, além de possibilitar que o administrador do sistema automatize várias de suas tarefas com o uso de *scripts*.

A história do Registro do Windows remonta a era do Microsoft DOS[1]. Segundo (HONEYCUTT, 2005), o MS-DOS usava os arquivos `Config.sys` e `Autoexec.bat` para configurar o SO durante o seu processo de inicialização. Assim, enquanto o `Config.sys` era reponsável por carregar *drivers* de dispositivos, o `Autoexec.bat` executava programas, configurava variáveis de ambiente e preparava o sistema para uso. Desta forma, cada aplicação do MS-DOS era responsável por gerenciar suas próprias configurações, o que contribuia para a inexistência de um padrão para arquivos de configuração de aplicações.

O Windows 3.0 trouxe, como uma de suas novidades, os arquivos INI, que eram responsáveis por armazenar configurações do SO e de programas do mesmo. Tais arquivos eram simplesmente arquivos de texto com uma sintaxe simples e, como cada programa tinha o seu próprio arquivo INI, o sistema ficava repleto de arquivos deste tipo. Além disso, como observa (HONEYCUTT, 2005), os arquivos INI não eram poderosos o suficiente para implementar relacionamentos mais complexos entre o *software* e o SO, não proviam um padrão para armazenar tipos similares de configurações e nem hierarquia.

Com o Windows 3.1, surgiu o conceito de registro como uma ferramenta para armazenar configurações OLE (*Object Linking and Embedding* – Ligação e Encaixe de Objetos) e posteriormente os Windows 95 e NT 3.5 o expandiram para o que ele se tornou atualmente: um banco de dados de configuração do SO e *soft-*

[1] Por ser amplamente conhecido como MS-DOS, este texto usará este nome para referenciar este sistema da Microsoft.

wares em geral. Apesar do Registro tornar dispensável o uso de arquivos INI, estes ainda encontram-se em uso devido a algumas aplicações (o próprio Windows XP os utiliza, como no arquivo `Win.ini`, por exemplo).

Para demonstrar a importância do Registro para o Windows, (HONEYCUTT, 2005) utilizou um *software* para gerenciar os acessos do SO ao próprio Registro e o deixou executando em segundo plano, enquanto realizava suas tarefas de praxe como usuário. Ao final dos testes, o autor verificou que, para cada ação que ele tomava, por menor que esta fosse, como um clique no mouse, o Windows fazia um acesso ao seu Registro, o que o levou a definir o Registro como coração e alma do Windows.

De fato, o Registro do Windows é peça fundamental para o funcionamento do sistema e talvez por isso seja tão difícil encontrar documentação de qualidade para a gerência do mesmo. (HONEYCUTT, 2005) critica a Microsoft pela a falta de informação sobre o Registro, tanto dentro do Windows quanto em guias e tutoriais e conclui que este ato da empresa contribui para a mitificação do assunto.

2.2.1 Estrutura

A estrutura do Registro do Windows é muito parecida com a estrutura do seu sistema de arquivos. Uma das diferenças já evidencia um conceito fundamental do Registro: o que no sistema de arquivos é conhecido como pasta ou diretório, no Registro é conhecido como chave. O Registro possui muitas chaves, mas todas derivam de um conjunto de cinco chaves, denominado Root Keys. Na tabela 2.1, pode-se observar a listagem destas cinco chaves com suas respectivas abreviaturas. Detalhes sobre cada chave serão descritos na seção 2.2.2.

Tabela 2.1: Root Keys e suas abreviaturas.

Nome	Abreviação
HKEY_CLASSES_ROOT	HKCR
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM
HKEY_USERS	HKU
HKEY_CURRENT_CONFIG	HKCC

A utilização das abreviaturas não é obrigatória, porém é mais comum encontrar referências às abreviaturas do que ao nome das chaves. Seguindo esta linha de

pensamento, este texto usará apenas as abreviaturas para se referir às Root Keys, em vez dos seus nomes completos.

O nome de uma chave está limitado a 256 caracteres Unicode[2] e qualquer caractere ASCII pode ser usado, com exceção de contrabarra (\), asterísco (*) e ponto de interrogação (?). Um caminho pode ser indicado no Registro de forma similiar com que é indicado no Windows, através do uso de contrabarras para separar chaves, como no exemplo `HKLM\SYSTEM\CurrentControlSet\Hardware Profiles\`.

Outro conceito tão importante quanto o de chaves, é o de valores. Se traçarmos uma analogia entre chaves e pastas, valores vão corresponder aos arquivos. Assim, pode-se afirmar, no escopo do Registro, que uma chave é um contêiner de valores. Um valor é composto por três elementos:

- **Nome**: segue as mesmas regras indicadas para as chaves e não possui extensão;
- **Tipo**: é o que determina o conteúdo do arquivo;
- **Dado**: pode ser de três tipos: vazio, nulo ou igual à informação armazenada.

Na tabela do apêndice A, são apresentados todos os tipos de dados disponíveis no Registro do Windows, bem como uma descrição sucinta sobre cada um deles.

2.2.2 Root Keys

A seguir, seguem breves descrições sobre cada uma das Root Keys, como descrito por (HONEYCUTT, 2005).

`HKEY_USERS`: Esta chave contém subchaves que armazenam informações gerais sobre as contas de usuários cadastradas no sistema.

`HKEY_CURRENT_USER`: Link para `HKU\SID`, onde `SID` é o identificador de segurança do console do usuário. Neste ponto torna-se conveniente definir o que é SID.

De acordo com (HONEYCUTT, 2005), SID é o acrônimo para *Security Identifiers* (Identificadores de Segurança) que é uma sequência de números gerada pelo

[2]De acordo com (Unicode Consortium, 2009), Unicode é um método criado para representar caracteres no computador, independente de programas ou linguagens.

LSA (*Windows Local Security Autorithy* – Autoridade de Segurança Local do Windows), que é única dentro do *host* e do domínio. Além disso, um número SID não se repete mesmo que a fonte que ele representa seja apagada. Por exemplo, para cada conta de usuário é criado um SID e mesmo que a conta seja apagada, o SID persiste no sistema. Caso a conta apagada seja criada novamente, ela receberá um novo SID. Na prática, o SID é, no sistema, como um CPF para um cidadão brasileiro: único e capaz de identificar o indivíduo em qualquer parte do país.

Apesar de aleatória, a criação de um SID segue um padrão bem definido. De acordo com (HONEYCUTT, 2005), um SID sempre começa com um `S-`, seguido da versão, que normalmente é igual a 1. A seguir um número indica o identificador de autoridade, que, para contas de usuários, é normalmente igual a 5, que corresponde à Autoridade NT. Após, há o identificador de domínio, que vai até 500, seguido por uma sequência que pode indicar, tanto a conta quanto o grupo daquele usuário.

`HKEY_LOCAL_MACHINE`: Contém configurações da máquina local, assim todas as operações que são realizadas nas subchaves desta Root Key são válidas para o *host*, o que inclui todos os usuários que o utilizam.

`HKEY_CLASSES_ROOT`: O primeiro tipo de informação que esta chave armazena são associações entre diferentes tipos de arquivos e os programas que os executam. O segundo tipo é uma classe de registros para objetos COM (*Component Object Model* – Modelo de Componente de Objeto). Normalmente esta chave é a que consome o maior espaço de armazenamento no Registro.

`HKEY_CURRENT_CONFIG`: um link para dados de configuração do perfil corrente de hardware em `HKLM\SYSTEM\CurrentControlSet\Hardware\Profiles\Current`, que por sua vez é outro link para `HKLM\SYSTEM\CurrentControlSet\Hardware\Profiles\NNN`, onde `NNN` é um número incremental que começa por 0000.

2.2.3 Manipulação do Registro

Segundo (HONEYCUTT, 2005), o usuário altera o Registro a todo momento enquanto usa o sistema, mas de forma indireta, através do SO. Usando o Editor do Registro, o usuário pode manipular o Registro sem uma interface, de forma direta. Isto pode ser excelente para administradores que conhecem o sistema e querem automatizar tarefas e implementar Políticas de Segurança, mas é perigoso, pois pode danificar o sistema inteiro com a execução de uma única instrução ou *script*.

Como observa (HONEYCUTT, 2005), nos Windows XP e 2003 ocorreram melhorias substanciais no Registro, se comparados às versões anteriores, com destaque para o melhor desempenho e suporte a tamanhos maiores do Registro, possibilitando que o mesmo seja tão grande quanto o espaço disponível em disco.

O Editor do Registro é comumente chamado por Regedit, que é uma simplificação do seu nome em inglês (*Registry Editor*). Por isso, Editor do Registro e Regedit referem-se ao mesmo *software*. Como o nome Regedit é mais simples e fácil de assimilar, este texto utilizará esta nomenclatura para designar tal programa.

Para executar o Regedit, basta clicar em `Iniciar/Executar` e digitar `regedit`. Como este aplicativo se encontra no *path* do sistema, não há a necessidade de se digitar exatamente sua localização. A interface do programa é bastante simples e parecida com o Windows Explorer. A lógica é a mesma: chaves e sua hierarquia são dispostas do lado esquerdo da tela (do ponto de vista do usuário) e, do lado direito, são listados os valores que se encontram dentro da chave selecionada no momento. Assim como no Windows Explorer, o usuário pode navegar entre chaves clicando sobre elas e expandir aquelas que possuirem subchaves, clicando no sinal de soma associado àquela chave. A figura 2.1 apresenta a interface gráfica com o usuário do Regedit.

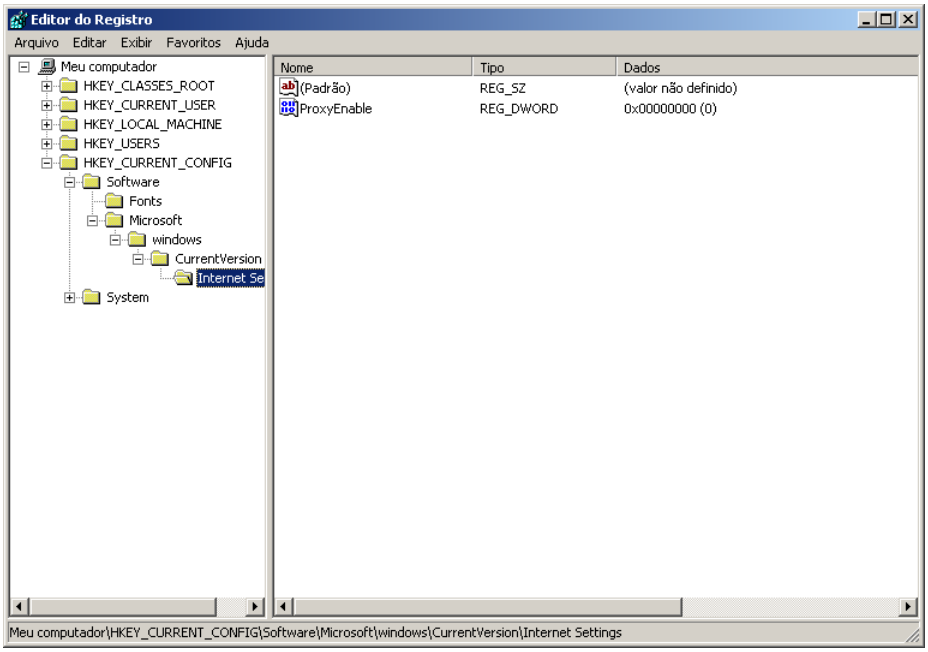

Figura 2.1: Janela do Regedit.

Como qualquer alteração direta no Registro pode causar danos irreparáveis no sistema é recomendável que seja feito um *backup* do mesmo, para que possa haver recuperação em caso de erro. O Regedit possui a função de exportar todo o Registro, gravando-o em um arquivo que pode ser importado mais tarde, para recuperação do sistema. Para ter acesso a esta função, basta acessar a opção `Arquivo/Exportar` dentro do Regedit. Na janela que se abrir, deve-se definir um nome para o arquivo a ser exportado e garantir que, no campo `Intervalo de Exportação`, esteja marcada a opção `Tudo`. Então deve-se clicar no botão `Salvar`, para concretizar a exportação. Caso ocorra algum problema em uma alteração inadequada do Registro, o usuário poderá importar o arquivo gerado no Regedit, que é um procedimento que pode ser realizado de forma análoga ao descrito.

Apesar deste artifício de *backup* funcionar para a maioria dos casos, pode ser que uma alteração no Registro inviabilize a importação dos dados, como a negação do acesso ao Regedit por parte do usuário. Por isso a melhor forma de evitar problemas é ter cautela ao se alterar qualquer informação dentro do Registro, pois isto pode tornar o SO inutilizável pelo usuário.

O processo para importação de *scripts* para o Regedit é similar ao de exportação. Basta abrir o Regedit, acessar a opção `Arquivo/Importar` e localizar o arquivo com o *script* a ser importado. Dependendo da configuração do sistema, ainda é possível importar um *script* com um duplo clique do mouse sobre o arquivo. A forma que este trabalho utiliza para automatizar esta importação é através da interface em modo texto do Regedit, onde chama-se o programa passando como argumento o caminho do *script* a ser importado, e.g., `regedit arquivo.reg`.

2.2.4 Criando *Scripts* para o Regedit

A criação de *scripts* ajuda o administrador a executar tarefas rotineiras de forma mais fácil e com menores chances de erros durante a execução. Não é necessário qualquer *software* adicional para criar um *script* para o Regedit do Windows XP, pois o mesmo pode ser criado em qualquer editor de texto, como o Bloco de Notas do próprio Windows ou o GEdit do ambiente gráfico GNOME, este para Linux.

A primeira linha do *script* sempre deverá indicar qual versão do Regedit será usada para interpretar o *script*. No Windows XP há as versões 4 e 5 disponíveis para uso. Caso o usuário resolva usar a versão 4, a primeira linha do *script* deverá ser igual a:

```
REGEDIT4
```

Caso a versão escolhida seja a 5, deve-se iniciar o *script* com:

```
Windows Registry Editor Version 5.00
```

A única diferença que o autor encontrou com relação a estas duas versões, além da citada, é com relação aos tipos `REG_EXPAND_SZ` e `REG_MULTI_SZ`. Na versão 4 do Regedit, cada caractere é armazenado numa lista com os valores em hexadecimal (seguindo a tabela ASCII) de cada um, separados por vírgula e no final há um valor nulo (`0x00`). Já para a versão 5 a lógica é a mesma, porém há um nulo entre cada caractere e a lista termina com dois nulos (na verdade são três: um nulo que entra naturalmente após o último caractere e dois pra concluir a lista).

Depois de indicar qual versão do Regedit será usada, as chaves e valores que serão alterados já podem ser declarados com seus novos dados. Para tanto, deve-se indicar o caminho do item a ser alterado entre colchetes (`[]`) e na linha seguinte, se for o caso, indicar os valores a serem alterados, um em cada linha, como no seguinte padrão:

- os nomes dos valores deverão estar entre aspas duplas (`"`);
- deve haver um sinal de igual logo a seguir, sem espaços;
- este ponto varia de tipo para tipo. Para o `DWORD`, por exemplo, o nome `dword` deve seguir o sinal de igualdade, também sem espaços e seguido por dois pontos (`:`). Logo depois, o valor que a variável receberá deve ser declarado, também sem que haja espaço entre os dois pontos e o valor. Já para uma *string* (`REG_SZ`), basta colocar a *string* entre aspas duplas e garantir que não haja espaços entre o sinal de igual e as aspas.

Na figura 2.2 há um trecho de código válido para o Regedit, explicitando as orientações anteriores.

```
1 [HKEY_CURRENT_CONFIG\Software\Fonts]
2 "FIXEDFON.FON"="vgafix.fon"
3 "FONTS.FON"="vgasys.fon"
4 "OEMFONT.FON"="vga850.fon"
5 "LogPixels"=dword:00000060
```

Figura 2.2: Trecho de *script* válido para o Regedit.

Há vários sites na Web que possuem informações sobre como construir *scripts* para o Regedit, além de apresentarem dicas de configuração de chaves para o

mesmo. Bons exemplos de sites assim são `http://technet.microsoft.com/pt-br/library/cc750583(en-us).aspx`, `http://support.microsoft.com/kb/310516/en-us` e o website `http://msdn.microsoft.com/en-us/library/ms724871(VS.85).aspx` (sites acessados em 1 de agosto de 2009). A seguir, são listados alguns detalhes relevantes de criação deste tipo de *script*.

1. Arquivos que contém *scripts* para o Regedit possuem a extensão `.reg`.

2. Cada chave que o usuário criar receberá um valor padrão que é como criar um diretório e o mesmo já vir com um arquivo dentro. Para uma determinada aplicação pode ser necessário alterar este valor, que é identificado pelo símbolo de arroba (`@`). Então, para alterar o seu valor, basta atribuir o novo valor ao símbolo de arroba, e.g., `@="Novo valor"`.

3. Para apagar uma chave via *script*, basta inserir o sinal de subtração (-) entre os colchetes, antes do caminho da mesma, e.g., `[-HKEY_LOCAL_MACHINE\SYSTEM\Setup]` (apagará a chave `Setup`).

4. Para apagar um valor, basta fazer com que o mesmo receba um sinal de subtração, e.g., `"FONTS.FON"=-`.

5. A contrabarra é um caractere de escape. Assim, caso seja necessário indicar um caminho do Windows para um valor, como ao definir um papel de parede, deve-se colocar duas contrabarras, e.g., `"Logon"="C:\\Windows\\Wallpaper.bmp"`.

6. Comentários no código podem ser feitos atráves do uso de ponto e vírgula (`;`), sendo que o editor, ao encontrar este caractere, ignora todo o texto até o final da linha. Funciona como o sustenido (`#`) em *shell script*.

Um exemplo de *script* funcional para a versão 4 do Regedit é apresentado na figura 2.3[3].

[3] As figuras deste texto apresentam quebras de linhas nas declarações de chaves do Regedit. Estas quebras têm efeito apenas estético, uma vez que o Regedit não as aceita. Assim, caso o leitor resolva testar um exemplo aqui apresentado, que contenha uma quebra de linha na declaração da chave, ele deve ignorá-la, colocando toda a declaração na mesma linha.

```
REGEDIT4

; Autor: José Lopes de Oliveira Júnior

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Internet Settings]
"DisablePasswordCaching"=dword:00000001

[HKEY_CLASSES_ROOT\regfile\shell]
@="edit"

[HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\
WindowsUpdate\AU]
"NoAutoUpdate"=dword:00000001
```

Figura 2.3: *Script* funcional para o Regedit.

2.3 Samba

De acordo com (HERTEL, 2001), Samba é um pacote de *software* disponível sob a licença pública GNU, usado para permitir que SOs que executam em plataformas Unix, como Linux e OS X, tenham acesso a compartilhamentos Windows.

Segundo (NEMETH; SNYDER; HEIN, 2007), na década de 1980 a IBM desenvolveu o NetBIOS (*Network Basic Input/Output System* – Sistema Básico de Entrada e Saída de Rede) como uma API (*Application Programmer's Interface* – Interface do Programa de Aplicação) que permitia que computadores na mesma sub-rede conversassem um com o outro usando nomes em vez de números. O NetBIOS logo foi incorporado à sua camada de transporte de rede subjacente, com o intuito de fazer os seus pacotes passarem através de redes Token Ring e Ethernet. Esta nova implementação recebeu o nome de NetBEUI (*NetBIOS Extended User Interface* - Interface de Usuário Estendida NetBIOS). Assim a API do NetBIOS tornou-se popular, sendo adaptada para uso na parte superior de vários protocolos de rede, inclusive o TCP/IP. Mais tarde foi criado, pela Microsoft em conjunto com a Intel, um protocolo de compartilhamento de arquivos sobre o NetBIOS, o qual foi denominado SMB (*Server Message Block* - Servidor de Bloco de Mensagens). Quando o SMB tornou-se capaz de operar em redes remotas, seu nome mudou para CIFS (*Common Internet File System* - Sistema de Arquivos Comum da Internet).

Com a disseminação de aplicações que usavam o protocolo NetBIOS, logo surgiu a necessidade de usuários de outros SO's fazerem uso das mesmas. Um

destes usuários foi o australiano Andrew Tridgell. A princípio seu problema consistia apenas em montar um disco rígido de um servidor Unix em um computador com o MS-DOS. Este foi facilmente resolvido com o uso de um cliente NFS (*Network File System* - Sistema de Arquivos de Rede) no MS-DOS. Contudo, Andrew ainda tinha uma aplicação que usava a interface NetBIOS e (HERTEL, 2001) deixa a entender que usar múltiplos protocolos no MS-DOS era sinônimo de problema.

Assim, Andrew escreveu um *sniffer* de pacotes, fez engenharia reversa no protocolo SMB e o implementou no Unix. Isto possibilitou que ele montasse sistemas de arquivos compartilhados no Unix enquanto executava aplicações NetBIOS. Posteriormente, Andrew publicou este código e houve algumas correções de *bugs*. Isto ocorreu em 1992. Dois anos mais tarde, Andrew decidiu fazer seu computador com Linux comunicar-se com o da sua esposa, que rodava Windows. Sem muitas opções para resolver o problema, ele recorreu ao código que havia escrito anos antes e o mesmo funcionou, surpreendendo-o.

Andrew logo descobriu que, àquela época, tanto o NetBIOS quanto o SMB estavam documentados. Isto o incentivou a voltar ao trabalho, mas um outro problema logo apareceu: havia uma empresa requerendo os direitos sobre o nome SMB, que era o nome do seu projeto até então. Por isso, Andrew teve de procurar por um nome alternativo. Com a ajuda de um verificador ortográfico, ele procurou por palavras que continham as letras `S`, `M` e `B`. Assim surgiu o Samba.

O Samba oferece, através do CIFS, cinco serviços básicos (NEMETH; SNYDER; HEIN, 2007):

- compartilhamento de arquivos;
- impressão em rede;
- autenticação e autorização;
- conversão de nomes;
- anúncio de serviços ("navegação"de impressoras e de servidor de arquivos).

Além desses serviços, o Samba também pode ser usado como um controlador de domínio primário do Windows, também conhecido como PDC, que é a sigla para *Primary Domain Controller*. (HAYDAY, 2001) define um PDC como a autoridade de segurança central de um domínio Windows NT. Ainda segundo esta fonte, uma rede Windows NT deve ser constituída de, pelo menos, um PDC, com zero ou mais clientes e, opcionalmente, um BDC (*Backup Domain Controller* –

Controlador de Domínio de *Backup*), que não passa de uma cópia de segurança do banco de dados de senhas dos usuários cadastrados no PDC, com a finalidade de tomar o lugar deste no domínio em caso de falha.

Neste trabalho o Samba será usado como PDC, centralizando todos as contas de usuários do domínio e o mesmo ainda será responsável por armazenar o *script* de configuração que será executado quando um cliente entrar no domínio. Desta forma, a configuração do Samba é essencial para o desenvolvimento deste projeto.

2.3.1 Configuração do Samba

O arquivo responsável por armazenar as configurações do Samba é o `smb.conf` e sua localização pode variar de instalação para instalação, mas normalmente ele fica armazenado em `/etc/samba` e pode ser manipulado por qualquer editor de textos como o vi, por exemplo.

A sintaxe de configuração do arquivo `smb.conf` não é muito complexa. Consiste basicamente de seções que possuem seus nomes entre colchetes, parâmetros com atribuições de valores para estes e comentários, definidos com "`;`"ou "`#`". Neste arquivo ainda podem ser usadas algumas variáveis, de acordo com a tabela 2.2.

A primeira seção do `smb.conf` é a `[global]` e, para o desenvolvimento deste trabalho, os seguintes parâmetros merecem especial atenção:

`workgroup`: de acordo com (ECKSTEIN; COLLIER-BROWN; KELLY, 1999), este parâmetro configura o grupo de trabalho ou o domínio onde o servidor Samba se anunciará. O valor deste parâmetro, assim como o próximo (`netbios name`), deverá seguir as regras de nomes suportados pelo protocolo NetBIOS: até 15 caracteres (`a-Z`, `0-9`, `!`, `@`, `#`, `$`, `%`, `^`, `&`, `(`, `)`, `-`, `'`, `{`, `}`, `.`, `~`), sendo que (ECKSTEIN; COLLIER-BROWN; KELLY, 1999) não recomenda o uso do ponto final, por questões de compatibilidade com futuras versões do protocolo;

`netbios name`: permite ao administrador configurar o nome NetBIOS do servidor, de acordo com as regras já citadas no item anterior. O valor padrão para esta variável é igual à variável de ambiente `$HOSTNAME` do servidor;

`server string`: define um comentário para ajudar a identificar o servidor no domínio;

Tabela 2.2: Variáveis do Samba.

Variável	Descrição
Clientes	
%a	Arquitetura do *host*.
%l	IP do *host*.
%m	Nome NetBIOS do *host*.
%M	Nome DNS do *host*.
Usuários	
%u	Nome do usuário corrente (*Unix*).
%U	Nome de usuário da sessão.
%H	Diretório *home* de %u.
%g	Grupo primário de %u.
%G	Grupo primário de %U.
Compartilhamentos	
%s	Nome do compartilhamento corrente.
%p	Caminho do diretório *home* do serviço, obtido de sua entrada NIS no `auto.map`. A entrada NIS no `auto.map` é expandida como `%N:%p`.
%P	Diretório *home* do compartilhamento atual.
Servidores	
%d	ID corrente do processo servidor.
%h	Nome DNS do servidor Samba.
%L	Nome NetBIOS do servidor Samba.
%N	Diretório *home* do usuário fornecido por %u.
%v	Versão do Samba.
Miscelânea	
%R	O nível que foi negociado do protocolo SMB.
%T	Data e hora atuais.
%$VAR	Valor da variável de ambiente `VAR`.

`security`: define o modo de segurança para negociações cliente-servidor. O valor padrão pode ser usado, que é igual a `user`, o que requer cadastro do usuário no servidor para que o mesmo possa entrar no domínio;

`encrypt passwords`: garante o tráfego de senhas criptografadas pelo domínio quando configurado com o valor `Yes`;

`domain logons` & **`domain master`**: juntos, estes dois parâmetros são parte fundamental da configuração do servidor como PDC ou BDC. De acordo com (Samba Team, 2008), quando os dois parâmetros são iguais a `Yes`, o servidor

será configurado como PDC. Se o primeiro for igual a `Yes` e o segundo for igual a `No`, o servidor será um BDC;

`local master`, **`preferred master`**, **`os level`**: complementam a configuração do servidor como PDC quando `local master` e `preferred master` são iguais a `Yes`. `os level` é um parâmetro usado pelo Windows para eleger um controlador de domínio. O sistema com o maior número vence a eleição e torna-se o controlador. (ECKSTEIN; COLLIER-BROWN; KELLY, 1999) descreve o valor 65 como suficiente para que o servidor vença todas as máquinas Windows;

`logon path`: especifica o diretório onde perfis remotos estão armazenados (Samba Team, 2008);

`logon home`: informa onde se localiza o diretório *home* quando uma estação entra no domínio (Samba Team, 2008);

`logon drive`: é usado apenas por clientes NT. Especifica o caminho local para o diretório *home* (Samba Team, 2008);

`logon script`: especifica um arquivo de *script* (`.bat` ou `.cmd`) para ser executado quando um cliente entra no domínio. É importante notar que tal arquivo deve conter o formato de quebra de linha do Windows (`CR/LF`) (Samba Team, 2008).

Além da seção `[global]`, a seção `[netlogon]` deverá ser configurada, para que o Samba execute *scripts* no início ou término da sessão do usuário. Assim, quando um usuário entrar ou sair do domínio, o Samba executará a configuração da sua conta de acordo com parâmetros a serem definidos pelo administrador do sistema (e.g., dar permissões para uso do SO de acordo com o grupo do usuário logado). Para isso, os seguintes parâmetros da seção `[netlogon]` deverão ser configurados:

`path`: informa a localização do compartilhamento `[netlogon]` no sistema, que é onde normalmente residem os *scripts* de logon/logoff (ECKSTEIN; COLLIER-BROWN; KELLY, 1999);

`root preexec`: indica um comando a ser executado como root antes do estabelecimento da conexão (ECKSTEIN; COLLIER-BROWN; KELLY, 1999);

`root postexec`: idêntico ao anterior, mas o comando é executado ao final da conexão com o servidor (ECKSTEIN; COLLIER-BROWN; KELLY, 1999).

2.3.2 Configuração das Contas de Usuários e Máquinas

A primeira etapa de configuração de contas é a criação da senha para o usuário `root` do Samba. O comando a seguir permitirá que o administrador crie esta conta:

```
# smbpasswd -a root
```

A opção `-a` indica que a senha será cadastrada no banco de dados local do `smbpasswd`. Com a senha da conta root configurada, as contas de usuários deverão ser cadastradas. Para isto, as contas deverão ser criadas dentro do SO e depois dever-se-á configurar a senha para cada conta no Samba (sendo esta última etapa necessária apenas para usuários que precisarem se logar nos clientes Windows):

```
# adduser NOME
# smbpasswd -a NOME
```

A primeira linha de comando adiciona a conta de usuário no SO com nome de usuário igual a `NOME`[4]. Caso o administrador queira aumentar a segurança, o parâmetro `-s` acrescido da opção `/bin/false` pode ser adicionado ao comando para garantir que o usuário não possuirá um *shell* válido, o que aumenta a segurança do sistema. A segunda linha de comando cria uma senha para a conta recém criada dentro do Samba.

Para se cadastrar as máquinas que poderão se logar no sistema, os três comandos da figura 2.4 deverão ser digitados:

```
1 # useradd -s /bin/false -c "Dominio Grunge" -d /dev/null HOST$
2 # passwd -l HOST$
3 # smbpasswd -a -m HOST
```

Figura 2.4: Sequência de comandos para adição de máquinas no domínio.

No primeiro comando, o parâmetro `-s` com a opção `/bin/false` desabilita o *shell* da conta cadastrada. `-c` indica uma *string* a ser adicionada à conta como comentário. O parâmetro `-d` permite especificar o diretório *home* para a conta a ser adicionada no SO. Com a opção `/dev/null`, o administrador evita a criação deste diretório, que é desnecessário para as contas das máquinas. É interessante

[4]Este será o nome com o qual o usuário entrará no domínio.

notar que ao final do nome cadastrado na conta (HOST[5]) há um cifrão ($), que serve para indicar que uma conta para máquina está sendo criada. No segundo comando garante-se o travamento da conta recém criada com a opção -l para o passwd. No último comando adiciona-se a conta de máquina ao Samba. A opção -m indica a conta de máquina e não há a necessidade de se informar o cifrão no final do nome cadastrado.

Conforme (SILVA, 2007), o cadastro de contas de máquinas no servidor pode ser automatizado com a configuração do parâmetro add machine script no arquivo smb.conf. Após a configuração deste parâmetro, as máquinas poderão ser adicionadas com o logon da conta de administrador do Samba na própria máquina a ser cadastrada. Um exemplo de configuração para este parâmetro, que se localiza na seção [global] do arquivo de configuração, pode ser visto na figua 2.5.

```
1 # add machine script = useradd -s /bin/false
2 -c "Dominio Grunge" -d /dev/null %u
```

Figura 2.5: Automatização do cadastro de contas de máquinas no Samba.

2.3.3 Configuração dos Clientes

A configuração dos clientes (máquinas Windows) consiste em preparar o sistema para usar o domínio do Samba. Para isto, deve-se, em cada sistema Windows do domínio, configurar o nome da máquina e do domínio. De acordo com (Microsoft Corporation, 2005), a primeira coisa a fazer é acessar a máquina com a conta de administrador local da mesma. As configurações de nomes do *host* podem ser editadas através de Painel de Controle/Sistema/Nome do Computador/Alterar. Na janela que será aberta será necessário preencher o nome do computador[6] (se ainda não estiver configurado) e selecionar a opção Domínio no campo Membro de. Assim, deve-se digitar o nome do domínio, que deve ser igual àquele cadastrado no parâmetro workgroup do smb.conf. Então deve-se clicar em OK. Ao ser solicitado, o usuário deverá digitar um nome e uma senha já cadastrados no domínio e clicar em OK. Se tudo ocorrer bem, bastará clicar em OK para concluir e clicar em OK novamente para fechar a janela de propriedades do sistema. Com a entrada do computador no domínio será mostrada uma mensagem de boas vindas e a máquina deverá ser reiniciada para que as alterações surtam efeito.

[5]Este nome deverá ser igual ao nome da máquina, configurado no Windows.
[6]Este nome deverá ser igual a algum nome de máquina cadastrado no Samba.

2.4 Python

De acordo com (BRUECK; TANNER, 2001), Python é um linguagem interpretada, orientada a objetos, que pode ser aplicada em uma ampla gama de tarefas. Além disso, está disponível sob os formatos binário e em código fonte, podendo ser usada de forma totalmente gratuita e livre, sem a necessidade de se pagar *royalties* a qualquer empresa. Segundo a mesma fonte, todas as plataformas de SO's são suportadas pela linguagem, incluindo Linux, OS X e Windows.

A linguagem, segundo (VENNERS, 2003), foi desenvolvida por Guido van Rossum no início da década de 1990, mas sua história começou dez anos antes, quando Guido começou a trabalhar no time de desenvolvimento de uma linguagem chamada ABC no Instituto de Pesquisa Nacional para Matemática e Ciência da Computação (CWI - sigla para *Centrum voor Wiskunde en Informatica*). De acordo com (VENNERS, 2003), a linguagem ABC almejava ser uma linguagem que poderia ser ensinada a usuários avançados de computador, que não fossem especializados em programação. Após alguns anos de trabalho, com a linguagem pronta, os principais desenvolvedores passaram a ensiná-la a cientistas que precisavam interagir com seus computadores e a partir do uso da linguagem por estas pessoas e do *feedback* das mesmas para a equipe de desenvolvimento, criou-se a ideia de se desenvolver uma outra linguagem com menos limitações que a primeira.

(VENNERS, 2003) continua a história citando que o projeto da ABC não obteve muito êxito por ser pouco ambicioso, fazendo com que Guido resolvesse mudar de projeto. Dessa forma, ele passou a participar do projeto Amoeba, ainda dentro do CWI, que visava a construção de um SO distribuído. Foi quando surgiu a necessidade de se criar uma linguagem de *script* dentro deste projeto e como Guido já possuia *know-how* suficiente de construção de linguagens de programação, graças ao projeto ABC, ele assumiu esta tarefa. Assim, como (VENNERS, 2003) narra, Guido começou a escrever a sua linguagem e colocou nela todos os pontos que considerava positivos da ABC e retirou os que considerava negativos. Então, pouco a pouco ele foi construindo a primeira versão de sua linguagem de programação, a qual viria chamar de Python por causa de um grupo humorístico da rede de TV BBC chamado "*Monty Python's Flying Circus*".

Em fevereiro de 1991, segundo (Python Software Foundation, 2008), Guido resolveu publicar a linguagem Python na USENET (um meio de comunicação onde usuários postam mensagens de texto em fóruns que são agrupados por assunto). Nesta época a linguagem completava cerca de três anos de uso no Amoeba,

com relativo sucesso. A partir daí, a linguagem ganhou novos colaboradores, que adicionaram à mesma várias funcionalidades, como ferramentas para programação funcional, suporte nativo a números complexos e sistema coletor de lixo da memória. Uma listagem completa de todas as mudanças entre as versões 0.9.0 e 2.5 RC1, pode ser encontrada no endereço `http://svn.python.org/view/*checkout*/python/trunk/Misc/HISTORY`.

Como pode ser observado no site `http://www.python.org/about/apps/`, com o passar dos anos a linguagem Python passou a ser adotada por várias organizações, para os mais variados propósitos. Dentre as entidades que fazem uso desta linguagem, vale destacar o Google (renomado site de busca) e a NASA (sigla para *National Aeronautics and Space Administration* - Administração Nacional de Aeronáutica e Espaço). Ainda neste site é possível ver relatos de casos de sucesso de aplicação da linguagem, sendo que o de Peter Norvig, diretor de pesquisa e qualidade do Google Inc., chama a atenção:

> *"Python tem sido uma parte importante do Google desde o início [da empresa] e continua assim enquanto o sistema cresce e evolui. Atualmente dúzias de engenheiros do Google usam Python e nós estamos procurando por mais pessoas com habilidades nesta linguagem."*

Este texto não objetiva ser referência para o aprendizado da linguagem Python, uma vez que o conteúdo referente à mesma é muito vasto. Assim, caso o leitor queira se aprofundar na linguagem, o autor sugere a leitura de (BRUECK; TANNER, 2001), como uma fonte abrangente de conhecimentos sobre a linguagem. O site oficial da linguagem também é uma ótima fonte de informação, agregando alguns tutoriais que ensinam de forma clara e objetiva os fundamentos da mesma (`http://www.python.org/doc/`).

Uma vez que o interpretador Python esteja instalado no sistema, fato que é comum nas maiores distribuições Linux, pode-se chamá-lo no terminal com a seguinte linha de comando:

```
$ python
```

Isto iniciará o prompt do interpretador Python, onde comandos da linguagem poderão ser inseridos como no Bash. Para se executar um *script* que já contenha a chamada para o interpretador Python no seu cabeçalho (primeira linha igual a

`#!/usr/bin/env python`) e permissão para execução, o processo é o mesmo como em qualquer *script*, como no exemplo a seguir, que executa o programa em Python, `python_program.py`:

```
$ ./python_program.py
```

O exemplo acima ainda evidencia o uso da extensão `.py` para arquivos de código-fonte Python. É de essencial importância que o leitor entenda que Python é sensível ao caso dos caracteres, ou seja, `Pearl Jam`[7] é diferente de `pearl jam`, que é diferente de `PEARL JAM`, que é diferente de `PeaRL JaM`.

2.4.1 Indentação

Python difere de muitas linguagens por não usar caracteres específicos para delimitação de blocos de código. Toda a definição destes blocos é feita através da própria indentação do código, o que tende a tornar o código mais limpo e elegante, já que obriga o programador a criar e usar um padrão de indentação.

```
1 if b > 9:
2     if b < 11:
3         a = 7
4 else:
5     a = 77
```

Figura 2.6: Exemplo de indentação em Python.

Na listagem da figura 2.6, as linhas 2 e 3 estão logicamente inseridas na estrutura `if` da linha 1. O `else` da linha 4 pertence ao `if` da linha 1, por causa da indentação e a linha 5 está dentro do `else`.

2.4.2 *Strings*, Listas e Funções

De acordo com (BRUECK; TANNER, 2001), há várias formas de se criar *strings* em Python. As duas mais básicas são através do uso de aspas simples e duplas, como nos exemplos da figura 2.7.

[7] Pearl Jam é o nome de uma banda grunge estadunidense e os nomes mencionados nos exemplos subsequentes desta seção, são títulos de suas músicas.

```
1 'Uma string.'
2 "Outra string."
3 "Errado...'
```

Figura 2.7: *Strings* em Python.

Outra forma muito útil para criar *strings* com mais de uma linha, é usar três aspas duplas seguidas para abrir e fechar a *string*. Este método pode ser observado na figura 2.8 e é muito utilizado para documentação de códigos.

```
1 print """Uma string
2 de mais de uma linha,
3 na linguagem Python!"""
```

Figura 2.8: *String* de múltiplas linhas em Python.

O código da figura 2.8, geraria uma saída igual à da figura 2.9. Vale ressaltar que uma string de múltiplas linhas, quando usada para documentar programas em Python, é chamada de *docstring*.

```
1 Uma string
2 de mais de uma linha,
3 na linguagem Python!
```

Figura 2.9: Resultado da execução do código da figura 2.8.

Assim como na linguagem C, é possível usar sequências de escape em *strings* para trabalhar a saída gerada pela mesma (veja a tabela 2.3):

Outro detalhe interessante sobre *strings* em Python é que as mesmas podem receber algumas operações de tipos numéricos, como no exemplo da figura 2.10.

```
1 'D' + 'i' + 's' + 's' + 'i' + 'd' + 'e' + 'n' + 't'
2 a = 'H'
3 a += 'ail-'
4 print a
5 a *= 2
6 print a
```

Figura 2.10: Operações com *strings* em Python.

Após a execução da linha 1, a palavra `Dissident` é impressa na tela. Na linha 2, a variável `a` recebe a string `H` e na linha 3, outra *string* é concatenada ao final

Tabela 2.3: Sequências de escape.

Sequência	Descrição
\n	Nova linha (ASCII LF).
\'	Aspa simples.
\"	Aspa dupla.
\ \	Contrabarra.
\t	Tabulação horizontal.
\b	Backspace (ASCII BS).
\r	Retorno de carro (ASCII CR).
\xHH	Caractere com o valor HH em hexadecimal.
\oHH	Caractere com o valor HH em octal.
\a	Campainha do sistema (ASCII BEL).
\v	Tabulação vertical (ASCII VT).

desta. Assim quando, na linha 4, esta variável é impressa, a *string* `Hail-` aparece na tela. Na linha 5, a variável `a` recebe seu conteúdo multiplicada por 2, fazendo com que após a linha 6 seja impresso `Hail-Hail-` na tela.

Como definido em (BRUECK; TANNER, 2001), uma lista é uma coleção ordenada de zero ou mais elementos de qualquer tipo de dados. Python é bastante flexível com relação a listas, o que torna a sua criação e manipulação bastante fácil, como mostrado na figura 2.11.

```
1 a = []  # Cria uma lista vazia, referenciada por a
2 b = ['Alive', 'Once', 'Footsteps']  # Cria uma lista de strings
3 c = [7, 'Given to Fly', b]  # Cria uma lista hibrida
```

Figura 2.11: Listas em Python.

Duas funções interessantes relacionadas a listas em Python são `list()` e `range()`. A primeira converte uma determinada sequência para uma lista e a segunda gera uma lista de números inteiros a partir de pontos de início e término estabelecidos, sendo que se o segundo parâmetro for omitido, será gerada uma lista com um número de elementos igual ao do parâmetro especificado, iniciando em 0. Exemplos dessas funções podem ser vistos na figura 2.12.

A definição de funções em Python é bastante simples, assim como a criação de *strings*.

No exemplo da figura 2.13, a palavra reservada `def` indica a criação de uma nova função. Em seguida, tem-se o nome da função, seguida da sua assinatura, que

```
1 list("You Are")  # Gera a lista ['Y','o','u',' ','A','r','e']
2 range(2, 7)  # Gera a lista [2, 3, 4, 5, 6]
3 range(7)  # Gera a lista [0, 1, 2, 3, 4, 5, 6]
```

Figura 2.12: Operações com listas em Python.

vem entre parênteses. Os dois pontos indicam que o corpo da função vem a seguir. Na linha 2 há um hábito comum para programadores Python documentarem suas funções: uma *string* declarada com dois pares de três aspas duplas. Em seguida, o corpo da função, que no caso do exemplo apenas imprime um *string* de várias linhas.

```
1  def help():
2      """Imprime a ajuda do programa na tela."""
3
4      # String de varias linhas
5      print """Uso: str2hex [OPCOES] TEXTO
6      Imprime na tela a string passada como parâmetro, na
7      forma hexadecimal.
8
9      -re4, --registry-editor4\t\tGera a saída compatível
10     com a versão 4 do registro.
11     -re5, --registry-editor5\t\tGera a saída compatível
12     com a versão 5 do registro.
13     -h, --help\t\t\tImprime a ajuda do programa na tela.
14     -V, --version\t\t\tImprime as notas da versão do
15     programa na tela.
16     """
```

Figura 2.13: Exemplo de função em Python.

No exemplo da figura 2.14, a assinatura da função na linha 1 declara e atribui à variável `strn` uma *string* vazia. Em seguida, a documentação da função no formato de *string* de várias linhas. Na linha 8 é criada uma lista vazia que será alterada dentro da função, para enviar o seu valor na linha 16 como o retorno da função.

```
1  def translate(strn=""):
2      """Converte cada caractere da string
3      strn, para sua representação em
4      hexadecimal.
5
6      """
7
8      ret = []  # Cria uma lista vazia
9
10     # Para cada caractere da string de entrada,
11     # adiciona-se o mesmo em notacao hexadecimal
12     # com o formado de dois digitos, na lista ret.
13     for s in strn:
14         ret.append("%02X" % ord(s))
15
16     return ret
```

Figura 2.14: Função em Python que converte os caracteres de uma *string* para seus valores em hexadecimal.

2.4.3 PyGTK

De acordo com (FINLAY, 2005), PyGTK é um conjunto de módulos que proveem uma interface Python para o GTK+ 2 [8].

Basicamente, um programa que utiliza o GTK+, implementa chamadas para *widgets*, que são elementos que compõem a interface gráfica, como caixas de texto, botões e barras de progresso. Então cabe ao programador gerenciar as interações do usuário com estes *widgets* através dos *signals* e dos *callbacks*, cujos conceitos serão apresentados na sequência.

Todo *widget* tem, associado a si mesmo, um ou mais *signals* que acusam quando um determinado evento ocorrer nele, como um clique do mouse ou o pressionar de uma tecla. Ao detectar a ocorrência de um *signal*, o programador pode determinar como o programa responderá a ele e é normalmente assim que se dá funcionalidade às interfaces. Esta resposta é feita por meio de funções que são associadas a um ou mais *signals* de um ou mais *widgets* e recebem o nome de *callbacks*.

Na listagem da figura 2.15, pode-se observar um exemplo de código em Python que utiliza o PyGTK. Nas linhas 13, 14 e 15 são feitas as importações dos módu-

[8] GTK+ é, segundo o site oficial do projeto (www.gtk.org), um *toolkit* para construção de interfaces gráficas.

los necessários para criação da interface gráfica. Na linha 17 é declarada a classe `HelloWorld`, que possuirá como atributos, todos os *widgets* usados na interface, as funções de *callback* e outras que dão funcionalidade à interface, como métodos.

Na linha 18 é declarado o método inicializador da classe e na linha 19, o atributo `window` recebe um *widget* `Window` do GTK+, responsável por criar uma janela padrão. Nas linhas 20 e 21 os *signals* `delete_event` e `destroy`, que são gerados quando o usuário tenta fechar a janela, são conectados, respectivamente, às funções de *callback* `delete_event` e `destroy`. Na linha 22 as bordas da janela têm seus valores alterados para 10 pixels.

Nas linhas 23 a 26, um *widget* do tipo `button` (botão) é criado, e ao *signal* `clicked`, que ele possui, são associados os *calbacks* `hello` (que imprime uma *string* na tela) e `gtk.Widget.destroy` (que o conecta ao *callback* `destroy` da janela). Finalmente, na linha 27 o botão é adicionado à janela e nas linhas 28 e 29 as funções de exibição da janela e do botão são executadas, garantindo que eles serão exibidos quando o programa for interpretado. Na linha 43, um peculiaridade da linguagem: um código pode ser executado, tanto como um programa, quanto como um módulo para outro programa.

Caso o mesmo seja executado como um programa, a variável `__name__`, inerente a qualquer código Python, tem seu valor igual a `__main__`. Nesse caso, a linha 44 será executada, com uma chamada para a classe `HelloWorld` e seu método `run`, responsável por inicializar o *loop* principal do GTK+, iniciando a interface gráfica.

Ao se executar este código, uma janela similar à da figura 2.16 será exibida. Nesta janela o usuário tem todas as funcionalidades básicas da maioria dos programas, como redimensionamento, minimização/maximização, movimentação e fechamento. No caso deste último é gerado um *signal* `delete_event`, que, por estar conectado ao *callback* `delete_event`, fecha a janela. O usuário também pode clicar no botão `Hello World`, que imprimirá na tela a *string* definida no método `hello` e fechará a janela, pelo fato do *signal* `clicked` do botão também estar associado ao *callback* `destroy` da janela (linhas 25 e 26).

```
#!/usr/bin/env python
# -*- coding: utf8 -*-

"""Implements simple operations on PyGTK.
Hello World is a simple Python program which
uses PyGTK to show a window with a Hello World
button. This window closes when button is clicked.
It's an adaptation of the original code in
PyGTK tutorial

"""

import pygtk
pygtk.require("2.0")
import gtk

class HelloWorld(object):
    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.connect("delete_event", self.delete_event)
        self.window.connect("destroy", self.destroy)
        self.window.set_border_width(10)
        self.button = gtk.Button("Hello World")
        self.button.connect("clicked", self.hello, None)
        self.button.connect_object("clicked",
                                   gtk.Widget.destroy,
                                   self.window)
        self.window.add(self.button)
        self.button.show()
        self.window.show()

    def run(self): gtk.main()

    def destroy(self, widget, data=None): gtk.main_quit()

    def delete_event(self, widget, event, data=None):
        return False

    def hello(self, widget, data=None): print "Hello World"

#
# Main
#
if __name__ == "__main__":
    HelloWorld().run()
```

Figura 2.15: Exemplo de uso do módulo PyGTK.

Figura 2.16: Resultado da execução do código da figura 2.15.

Capítulo 3

Desenvolvimento

Neste capítulo será mostrado como todas as ferramentas apresentadas no capítulo 2 foram combinadas para a realização do trabalho. Primeiramente será mostrado como o autor automatizou a criação dos *scripts* de configuração dos clientes Windows. Por último, serão descritas as etapas de configuração do servidor, de modo com que toda máquina logada no domínio fosse configurada de acordo com o grupo do usuário que se logou.

3.1 Criação dos *Scripts*

(SILVA, 2004) propôs uma solução semelhante à apresentada neste texto, mas como o enfoque dele era a configuração do servidor Samba para executar os arquivos de configuração do Regedit, pouca atenção foi dada à etapa de criação destes (sendo este o enfoque deste trabalho). Segundo a fonte, uma das principais referências sobre entradas para o Regedit era o site Winguides, que possuia várias destas entradas, todas bem descritas. Atualmente o Winguides chama-se PC-Tools (`http://www.pctools.com/guides/registry`) e possui, além de um vasto acervo de entradas para o Regedit, todas separadas por categorias, um guia bastante prático para iniciantes no assunto.

Assim como mostrado em (SILVA, 2004), a criação do arquivo para configuração das máquinas Windows é crucial para obtenção dos resultados esperados. Por isso, este foi o ponto de partida para realização do trabalho. A partir do PC-Tools, foram retiradas todas as entradas para o Regedit que o autor julgou pertinentes à aplicação. Uma vez selecionadas, as entradas foram transcritas para um

arquivo texto onde receberam, cada qual, um cabeçalho padrão para identificação, conforme o exemplo da figura 3.1.

```
1  ; Controle da Funcao de Auto-Execucao do CD-ROM
2  ;
3  ; DESCRIPTION
4  ;    Normalmente quando voce insere um disco no drive de CD-ROM,
5  ; o conteudo e executado automaticamente. Esta funcao lhe
6  ; permite desabilitar este comportamento.
7  ;
8  ; COMPATIBILITY
9  ;    Windows NT/2000/XP
10 ;
11 ; MODIFIED VALUES
12 ;    Autorun : dword : 00000000 = Desabilita a Auto-Execucao;
13 ; 00000001 = Habilita a Auto-Execucao.
14 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
15 [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CDRom]
16 "Autorun"=dword:00000000
```

Figura 3.1: Exemplo de entrada para o Regedit.

No exemplo da figura 3.1 pode-se observar que cada entrada cadastrada foi catalogada com uma descrição sucinta da mesma (linha 1), uma descrição detalhada explicando as implicações daquela entrada (linhas 3 a 6), as versões do Windows com as quais aquela entrada é compatível (linhas 8 e 9) e os valores alterados por aquela entrada (linhas 11 a 13). Em seguida é declarada a entrada em si (linhas 15 e 16), que são as linhas importadas pelo Regedit.

Na listagem da figura 3.2 pode-se observar outro exemplo de uma entrada catalogada pelo autor. Pode-se perceber que os campos de descrição sucinta (linhas 1 e 2), descrição detalhada (linhas 4 a 8), compatibilidade (linhas 10 e 11) e de valores modificados (linhas 13 a 15), estão todos presentes. Neste exemplo ainda há um campo extra contendo observações (linhas 17 a 19) sobre aquela entrada específica, tudo para facilitar a utilização destas entradas pelo usuário.

O autor percebeu que todo o trabalho de criação do *script* poderia ser passado para uma linguagem de programação, o que facilitaria futuras mudanças de parâmetros do *script*, como alteração de um valor, de ativado para desativado. Desta forma, a linguagem Python foi escolhida, primeiramente devido às excelentes referências que o autor tinha dela e também pela vontade que o mesmo tinha em aprendê-la. Por isso todo o *script* foi portado para a referida linguagem, fazendo com que cada entrada, como a da figura 3.1, se transformasse em uma função, como a da figura 3.3.

```
; Esconder as Configuracoes Painel de Controle, Impressoras e
; Conexoes de Rede
;
; DESCRIPTION
;    Esta restricao remove as configuracoes Painel de Controle,
; Impressoras e Conexoes de Rede do menu Iniciar. Se as
; configuracoes da Barra de Tarefas tambem estiverem escondidas,
; acarretara na completa remocao das configuracoes.
;
; COMPATIBILITY
;    Todos.
;
; MODIFIED VALUES
;    NoSetFolders : dword : 00000000 = Desabilita restricao;
; 00000001 = Habilita restricao.
;
; OBSERVATION
;    Esta restricao deve desabilitar tambem a hotkey do Windows
; Explorer (SUPER + E).
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Policies\Explorer]
"NoSetFolders"=dword:00000001
```

Figura 3.2: Outro exemplo de entrada para o Regedit.

A vantagem desta implementação é que as chamadas para cada função poderiam ser inseridas em qualquer programa Python, tornando o trabalho do administrador mais flexível, pois, se este quisesse alterar o retorno de uma função, bastaria alterar o parâmetro passado.

Ao criar esta solução, o autor percebeu que havia desenvolvido uma camada de abstração em Python para o Registro do Windows. Assim surgiu a ideia de reunir todas as funções criadas em um único módulo Python, módulo este que recebeu o nome de Windows Registry Fixer (WRF). O WRF foi liberado como software livre (sob a licença GPLv3) pelo autor e o site oficial do projeto foi hospedado no Google Code[1], podendo ser acessado pelo endereço `http://code.google.com/p/windowsregistryfixer/`. Neste site, o WRF poderá ser copiado como um arquivo compactado. Ao descompactá-lo, será criado um diretório chamado `wrf-VERSÃO` (e.g.,`wrg-0.1.0`). Dentro deste diretório haverá alguns arquivos, sendo que o principal é o `wrf.py`, que corresponde ao módulo WRF.

[1] Google Code (`http://code.google.com/`) é um serviço gratuito de hospedagem de projetos, similar ao SourceForge (`http://sourceforge.net/`).

```
1  def autorun(on=0):
2
3      """Controle da Funcao de Auto-Execucao do CD-ROM
4
5      DESCRIPTION
6          Normalmente quando voce insere um disco no drive de CD-
7      ROM, o conteudo e executado automaticamente. Esta funcao
8      lhe permite desabilitar este comportamento.
9
10     COMPATIBILITY
11         Windows NT/2000/XP
12
13     MODIFIED VALUES
14         Autorun : dword : 00000000 = Desabilita a Auto-Execucao;
15     00000001 = Habilita a Auto-Execucao.
16
17     """
18
19     if on:
20         return '''[HKEY_LOCAL_MACHINE\\SYSTEM\\
21 CurrentControlSet\\\Services\\CDRom]
22 "Autorun"=dword:00000001'''
23
24     else:
25         return '''[HKEY_LOCAL_MACHINE\\SYSTEM\\
26 CurrentControlSet\\\Services\\CDRom]
27 "Autorun"=dword:00000000'''
```

Figura 3.3: Função em Python correspondente a uma entrada para o Regedit.

No exemplo da figura 3.3, pode-se perceber que o cabeçalho da entrada descrita na figura 3.1 foi transcrito para uma *docstring* em Python (linhas 3 a 17), de forma a documentar a função criada. O leitor pode perceber que a função `autorun` recebe um valor para indicar se a autoexecução do CD-ROM será habilitada ou não e um `if` (linhas 19 a 27) determina a *string* que esta função retornará. Esta *string* de retorno corresponde a uma entrada para o Regedit e pode ser gravada em um arquivo que contém outras entradas para o Regedit, como forma de ativar ou desativar este comportamento específico do Windows.

Todas as outras entradas catalogadas receberam tratamento semelhante ao da figura 3.3, tornando-se funções Python com um valor de entrada (o parâmetro para a função), a *docstring* e o processamento da função. A listagem da figura 3.4 traz outro exemplo, similiar ao da figura 3.3. O leitor pode perceber que todos os elementos da figura 3.3 estão presentes, sendo as funções semelhantes. As únicas

diferenças ficam por conta do conteúdo da *docstring* e dos possíveis valores de retorno da função.

```
def no_set_folders(on=0):

    """Esconder as Configuracoes Painel de Controle,
    Impressoras e Conexoes de Rede

    DESCRIPTION
        Esta restricao remove as configuracoes Painel de
    Controle, Impressoras e Conexoes de Rede do menu Iniciar.
    Se as configuracoes da Barra de Tarefas tambem estiverem
    escondidas, acarretara na completa remocao das
    configuracoes.

    COMPATIBILITY
        Todos.

    MODIFIED VALUES
        NoSetFolders : dword : 00000000 = Desabilita restricao;
    00000001 = Habilita restricao.

    OBSERVATION
        Esta restricao deve desabilitar tambem a hotkey do
    Windows Explorer (SUPER + E).

    """

    if on:
        return '''[HKEY_CURRENT_USER\\Software\\Microsoft\\\
Windows\\CurrentVersion\\Policies\\Explorer]
"NoSetFolders"=dword:00000001'''

    else:
        return '''[HKEY_CURRENT_USER\\Software\\Microsoft\\\
Windows\\CurrentVersion\\Policies\\Explorer]
"NoSetFolders"=dword:00000000'''
```

Figura 3.4: Outra função em Python correspondente a uma entrada para o Regedit.

Pode-se perceber que o usuário do WRF teria que consultar a implementação do próprio módulo para saber quais as funções disponíveis e o que cada uma faria, para então inseri-la no seu programa gerador de *scripts* para o Regedit. A vantagem é que o usuário teria que saber apenas o nome e a descrição da cada função, sem ter que se preocupar com quais chaves ou valores do Registro seriam alterados.

Já a desvantagem é que ainda assim o administrador seria obrigado a conhecer uma linguagem (Python), além de ter que criar um programa para usar o WRF.

Na figura 3.5 pode-se observar como o WRF pode ser utilizado diretamente pelo administrador. Nas linhas 1 e 2 desta listagem são definidas, respectivamente, o interpretador e a codificação do arquivo Python. Na linha 4 é importado o módulo `wrf` e uma função responsável por gravar uma *string* em um arquivo é definida nas linhas 7 a 10. A principal função do programa é definida entre as linhas 12 e 28.

Na linha 15 é aberto um arquivo de texto para escrita, chamado `wrf.reg`, sendo o mesmo associado à variável `file`. Na linha 17, a versão do Regedit que será responsável por interpretar o código que será gerado, é escrita no arquivo. A linha 19 faz uma chamada à função `autorun`, descrita na figura 3.3, do WRF, habilitando a autoexecução do CD-ROM. As linhas 21 a 23 executam a função do WRF, `disallow_run`, que inibe a execução dos arquivos passados à ela na lista (`Virus.exe`, `Worm.exe` e `Trojan.exe`).

A linha 24 roda a função `no_set_folders`, descrita na figura 3.4, removendo os itens `Painel de Controle`, `Impressoras` e `Conexões de Rede` do menu Iniciar. As linhas 25 e 26 utilizam a função `wallpaper_image_and_style` para definir novas configurações de papel de parede. Na linha 28, o arquivo é fechado. Na linha 33 verifica-se se o código está sendo executado como um módulo ou como um programa e somente neste último caso, as linhas 34 e 35 são interpretadas, para, respectivamente, executar a função principal do código e indicar o término com sucesso do mesmo.

Ao ser executado o código da figura 3.5, um arquivo com conteúdo semelhante ao da figura 3.6 seria gerado. É interessante notar a correspondência entre as listagens da figura 3.6 e da figura 3.5. A linha 1 da figura 3.6 foi gerada pela linha 17 da figura 3.5. Nos mesmos moldes, as linhas 3 a 5 da figura 3.6 foram geradas pela linha 19 da figura 3.5. As linhas 6 a 14 e 16 a 18 da figura 3.6 foram geradas pelas linhas 21 a 23 e pela linha 24 da figura 3.5. Finalmente, as linhas 19 a 22 da figura 3.6 foram geradas pelas linhas 26 e 27 da figura 3.5.

Exemplos como o apresentado nas figuras 3.5 e 3.6 evidenciaram a necessidade de se criar uma interface para o WRF, que possibilitasse criar *scripts* de forma intuitiva, sem que fosse preciso conhecer a sintaxe do Regedit ou de qualquer outra linguagem.

Neste contexto foi criado o GTK Windows Registry Fixer (GWRF), uma interface para o WRF escrita em GTK+. Um exemplo do GWRF pode ser visto na

```
#!/usr/bin/env python
# -*- coding: utf8 -*-

import wrf


def save_into_file(file, string):
    """Saves string into file."""

    file.write(string + '\n')

def main():
    """Program's main function."""

    file = open("wrf.reg", "w+t")

    save_into_file(file, "REGEDIT4\n")  # Header

    save_into_file(file, wrf.autorun(1))

    save_into_file(file, wrf.disallow_run(1, ["Virus.exe",
                                              "Worm.exe",
                                              "Trojan.exe"]))
    save_into_file(file, wrf.no_set_folders(1))
    save_into_file(file, wrf.wallpaper_image_and_style(1,
                 "C:\\\\Windows\\\\Wallpaper-Logon.bmp", 2))

    file.close()


# Main
if __name__ == "__main__":
    main()
    exit(0)
```

Figura 3.5: Exemplo de utilização direta do WRF.

figura 3.7. O leitor pode observar na figura que o GWRF possui opções para ativar/desativar/configurar os 65 itens que compõem o WRF. O GWRF faz parte do projeto WRF desde sua versão 0.1.0, portanto ao baixar o WRF o GWRF já será copiado, bastando que se execute o arquivo `gwrf.py`, presente no mesmo diretório do WRF, para iniciá-lo.

```
REGEDIT4

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Policies\Explorer]
"NoRun"=dword:00000001
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Policies\Explorer]
"DisallowRun"=dword:00000001

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Policies\Explorer\DisallowRun]
"1"="Virus.exe"
"2"="Worm.exe"
"3"="Trojan.exe"

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Policies\System]
"NoDispSettingsPage"=dword:00000001
[HKEY_CURRENT_USER\Control Panel\Desktop]
"TileWallpaper"="1"
"Wallpaper"="C:\\Windows\\Wallpaper-Logon.bmp"
"WallpaperStyle"="2"
```

Figura 3.6: Saída da listagem 3.5.

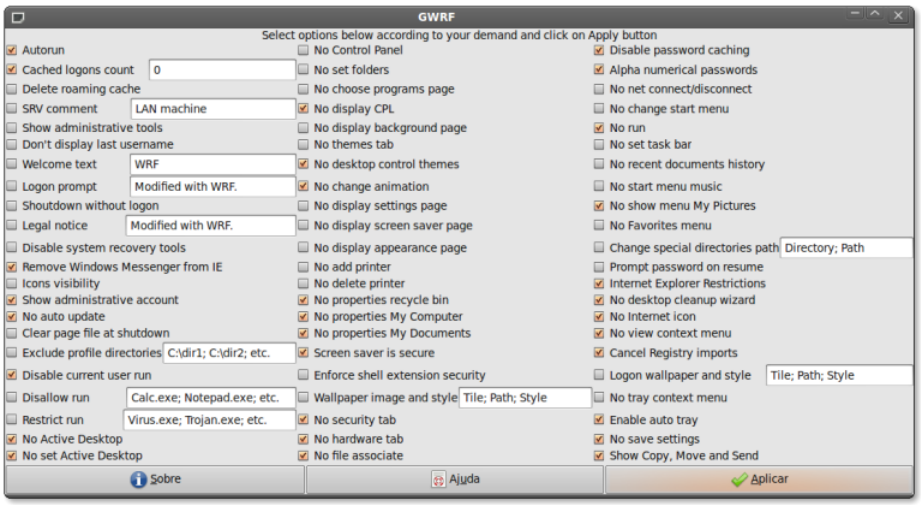

Figura 3.7: Captura de tela do GWRF.

Na listagem da figura 3.7, pode-se perceber que o GWRF é composto basicamente por um conjunto de *checkboxes*[2], que determinam as opções do usuário

[2]Um *checkbox* é um *widget* do GTK+ que possui a característica de poder ser marcado ou desmarcado pelo usuário, o que, neste contexto, indica se determinada função estará habilitada ou desabilitada.

para a criação dos *scripts*. No rodapé da janela há três botões: o botão `Sobre`, que abre a janela de créditos do programa; o botão `Ajuda` que abre o navegador padrão do sistema, no site do projeto WRF e o botão `Aplicar`, que gera o *script* a partir das opções selecionadas.

O uso do GWRF é bastante simples: o usuário seleciona quais opções devem ser ativadas, marcando cada caixa de seleção e configurando a opção quando for o caso e clica no botão `Aplicar`, que criará um arquivo chamado `wrf.reg` no diretório do próprio programa. Este arquivo possuirá todas as configurações selecionadas pelo usuário e prontas para serem importadas pelo Regedit.

Algumas mudanças feitas no Registro do Windows requerem reinicialização do sistema, outras não e mesmo o site PC-Tools não informa com exatidão este detalhe. Por isso o autor recomenda que, sempre que um *scripts* do Regedit for importado no sistema, que o mesmo seja reiniciado, para garantir que todas as mudanças surtam efeito.

Com o desenvolvimento do WRF e do GWRF, o autor chegou a um ponto onde a criação dos *scripts* para o Regedit foram limitadas à seleção de opções na tela, de forma gráfica, intuitiva para o usuário. Isto deve facilitar o trabalho dos administradores que fizerem uso da solução apresentada neste trabalho e em qualquer outra que necessite que *scripts* para o Regedit sejam criados, desde que as mesmas já estejam previstas nos softwares criados.

3.2 Servidor Samba

3.2.1 Introdução

Nesta seção serão apresentados os procedimentos para configuração do servidor Samba. Este servidor será usado como PDC no domínio Windows, além de armazenar os arquivos de configuração do Regedit, que serão responsáveis por configurar o sistema cliente, de acordo com o grupo do usuário logado.

3.2.2 Configuração

Para o servidor foi utilizada a distribuição Debian em sua versão 4.0 (Etch), juntamente com o Samba versão 3.0.24-6 (pacotes `samba`, `samba-common`, `smb-client`). Como as instalações destes *softwares* estão bem documentadas, além de serem

intuitivas, elas não serão abordadas. Então pressupõe-se que os *softwares* mencionados estejam instalados e funcionais, além do SO já ter sua interface de rede configurada de modo que seja possível acessar a rede à qual está conectado.

Para a configuração do Samba, deve-se alterar o arquivo `smb.conf`, ajustando corretamente os parâmetros descritos na sessão 2.3.1. Estes parâmetros devem ser configurados de forma a deixar o Samba como PDC e fazendo com que o mesmo execute os arquivos smblogon e smblogoff (descritos na sequência desta seção), quando os usuários entrarem ou sairem do domínio.

3.2.2.1 smblogon

Este é o *script* executado no logon do usuário, definido pela diretiva `root preexec` no arquivo `smb.conf`. Um exemplo de configuração desta diretiva pode ser observado na figura 3.8, linha 7. Já o conteúdo do smblogon pode ser visto na figura 3.9 e foi baseado em (SILVA, 2004).

```
1 [netlogon]
2     comment = Servico de Logon na Rede
3     path = /var/lib/samba/netlogon
4     write list = root
5     browseable = no
6     root preexec = /usr/sbin/smblogon %U %g %m %L
7     root postexec = /usr/sbin/smblogoff %g
```

Figura 3.8: Trecho do arquivo `smb.conf`, exemplificando a configuração da diretiva `root preexec`.

A primeira linha do smblogon indica o interpretador do *script* (Bash). Em seguida, são declaradas as variáveis usadas no *script* e a elas são atribuídos os parâmetros passados para o *script* na sua chamada na linha de comando (o leitor deve perceber que estes parâmetros são passados pela diretiva `root preexec` – linha 6 da figura 3.8 –, através das variáveis do Samba). Em seguida, são inicializadas as variáveis `LOGONBAT` e `LOGCMD` e, na linha 11, é feita uma marcação no arquivo de log do Samba para o logon do usuário. Nas linhas 13 e 14 é criada a chamada para o arquivo de configuração do Regedit, de acordo com o grupo do usuário recém logado. Este arquivo do Regedit pode ser gerado de forma automatizada pelo WRF em conjunto com o GWRF.

```
1  #!/usr/bin/env bash
2
3  # Variables
4  USER=$1
5  GROUP=$2
6  HOSTNAME=$3
7  SERVERNAME=$4
8  LOGONBAT="/etc/samba/netlogon/$GROUP.bat"
9  LOGCMD="logger -t SMB-PDC"
10
11 $LOGCMD "Login started => $USER@$HOSTNAME"
12
13 printf "%s%s%s%s%s \x0d\x0a" 'regedit /s \\' $SERVERNAME \
14 '\etc\samba\netlogon\' $GROUP '.reg' > $LOGONBAT
```

Figura 3.9: *Script* que será executado durante a entrada no domínio.

3.2.2.2 smblogoff

Se o smblogon é responsável por gerar o *script* de logon dos usuários, o smblogoff cria os *scripts* que são executados quando o usuário finaliza sua sessão no servidor. Na listagem da figura 3.10 pode-se observar um exemplo de *script* deste tipo, baseado em (SILVA, 2004).

```
1  #!/usr/bin/env bash
2
3  # Variables
4  GROUP=$1
5
6  rm /etc/samba/netlogon/$GROUP*.bat
```

Figura 3.10: *Script* que será executado durante a saída do domínio.

A linha 1 do *script* define o interpretador para o mesmo (Bash). Em seguida, é declarada uma variável e na linha 6 é executada a única operação deste *script*: ele remove o *script* criado pelo `smblogon` quando do início da sessão.

Capítulo 4

Resultados

Neste capítulo serão apresentados os resultados obtidos com a utilização do WRF. Primeiramente será apresentada a etapa de testes, onde serão descritos os testes realizados pelo autor para comprovar o funcionamento da solução proposta. Em seguida é apresentado, com a utilização de exemplos, como o WRF, através do GWRF, facilita a criação de scripts para o Regedit, tornando este processo transparente para o usuário.

4.1 Etapa de Testes

Para a realização dos testes, no servidor Samba foram criados os grupos `alunos` e `estagiários`. Em seguida foram criados os usuários `eddie` e `mike`, o primeiro pertencente ao grupo `alunos` e o segundo ao grupo `estagiarios`. Então o GWRF foi utilizado para criar os scripts de configuração dos dois grupos e os arquivos gerados foram depositados em `/etc/samba/netlogon`, sob os nomes de `alunos.reg` e `estagiarios.reg`.

A próxima etapa foi a realização do logon como usuário `eddie` e pôde-se perceber que todas as restrições impostas ao grupo `alunos` foram aplicadas. No servidor pôde-se observar a criação do arquivo `alunos.bat` dentro do diretório `/etc/samba/netlogon` (operação realizada pelo smblogon) e a posterior remoção deste arquivo no logoff.

Para o usuário `mike`, os resultados foram semelhantes, respeitando as diferenças com relação ao grupo. Em seguida, as permissões para o grupo alunos foram

alteradas e o arquivo `/etc/samba/netlogon/alunos.reg` foi novamente gerado (usando o GWRF) e atualizado no servidor. Com a nova versão do referido arquivo substituindo a anterior, foi feito um novo logon no domínio com o usuário `eddie` e percebeu-se que as novas alterações foram aplicadas com sucesso.

Na listagem da figura 4.1 pode-se observar alguns trechos do primeiro arquivo `alunos.reg`. O código em questão evidencia que a permissão para autoexecução do CD-ROM foi concedida a este grupo de usuários (linhas 1 e 2), que a exibição das opções de vídeo foi desabilitada (linhas 3 a 5), que a aba `Segurança` da janela de propriedades de arquivos e pastas foi escondida (linhas 6 a 8) e que as configurações de papel de parede foram alteradas (linhas 9 a 12).

```
1  [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CDRom]
2  "Autorun"=dword:00000001
3  [HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
4  Policies\System]
5  "NoDispSettingsPage"=dword:00000001
6  [HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
7  Policies\Explorer]
8  "NoSecurityTab"=dword:00000001
9  [HKEY_CURRENT_USER\Control Panel\Desktop]
10 "TileWallpaper"="1"
11 "Wallpaper"="C:\\Windows\\Wallpaper-Logon.bmp"
12 "WallpaperStyle"="2"
```

Figura 4.1: Trecho do *script* de configuração de contas de alunos.

No código da figura 4.2 o leitor encontra um trecho do arquivo `estagiarios.reg`. Nas linhas 1 e 2 pode-se perceber que a permissão para autoexecução do CD-ROM foi concedida a este grupo, além de lhes ter sido retirado o item `Executar` do menu Iniciar (linhas 3 a 5). Outras alterações feitas foram a permissão de acesso à aba `Segurança` (linhas 6 a 8), novas definições para o papel de parede (linhas 9 a 12) e a garantia de que os menus de contexto teriam as opções `Copiar para`, `Mover para` e `Enviar para` (linhas 13 a 23).

A listagem da figura 4.3 evidencia algumas mudanças feitas pelo segundo arquivo `alunos.reg`. É interessante notar que, diferentemente da versão original deste arquivo, a permissão para autoexecução do CD-ROM foi negada (linhas 1 e 2) e a exibição das opções de vídeo foi habilitada (linhas 3 a 5). Além disso, a importação de *scripts* para o Regedit foi desabilitada (linhas 6 e 7) e foi desabilitada a senha para a proteção de tela (linhas 8 a 10).

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CDRom]
"Autorun"=dword:00000000
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Policies\Explorer]
"NoRun"=dword:00000001
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Policies\Explorer]
"NoSecurityTab"=dword:00000000
[HKEY_CURRENT_USER\Control Panel\Desktop]
"TileWallpaper"="1"
"Wallpaper"="C:\\Windows\\Novo-Wallpaper.bmp"
"WallpaperStyle"="2"
[HKEY_CLASSES_ROOT\AllFilesystemObjects\shellex\
ContextMenuHandlers\Copy To]
@="{C2FBB630-2971-11d1-A18C-00C04FD75D13}"

[HKEY_CLASSES_ROOT\AllFilesystemObjects\shellex\
ContextMenuHandlers\Move To]
@="{C2FBB631-2971-11d1-A18C-00C04FD75D13}"

[HKEY_CLASSES_ROOT\AllFilesystemObjects\shellex\
ContextMenuHandlers\Send To]
@="{7BA4C740-9E81-11CF-99D3-00AA004AE837}"
```

Figura 4.2: Trecho do *script* de configuração de contas de estagiarios.

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CDRom]
"Autorun"=dword:00000000
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Policies\System]
"NoDispSettingsPage"=dword:00000000
[HKEY_CLASSES_ROOT\regfile\shell]
@="edit"
[HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\
Control Panel\Desktop]
"ScreenSaverIsSecure"=dword:00000000
```

Figura 4.3: Trecho do novo *script* de configuração de contas de alunos.

4.2 GWRF

O autor destacou três exemplos para apresentar como o GWRF, em conjunto com o WRF, facilitam a criação de *scripts* para o Regedit.

Para o primeiro exemplo, após executado o GWRF (dentro do diretório do WRF, basta executar o arquivo `gwrf.py`), foram marcadas as opções de acordo com a figura 4.4. Após clicar no botão `Aplicar`, dentro do próprio diretório do WRF foi criado o arquivo `wrf.reg`, com conteúdo similar ao da figura 4.5. Este arquivo já está pronto para ser importado pelo Regedit e caso esta importação seja feita, o sistema será configurado para habilitar a autoexecução do CD-ROM, manter `0` contas no *cache* de logon do `host`, deletar cópias em *cache* de perfis remotos e para alterar a descrição do computador na rede para `GWRF Test`. Neste exemplo, a correspondência entre as opções da figura 4.4 e as linhas de código da figura 4.5, são, respectivamente, `Autorun` (linhas 5 e 6), `Cached logons count` (linhas 7 a 9), `Delete roaming cache` (linhas 10 a 12) e `SRV Comment` (linhas 13 a 15).

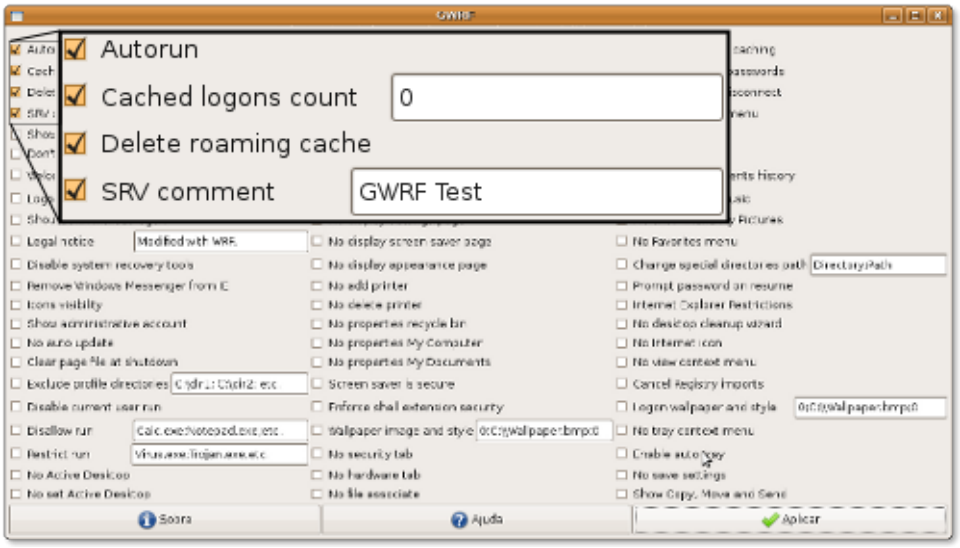

Figura 4.4: Exemplo de configuração no GWRF.

A figura 4.6 mostra a seleção de outras opções, que geram um arquivo similar ao da figura 4.7. É interessante notar a configuração para a imagem e o estilo do papel de parede do sistema, pois a mesma deve seguir a ordem indicada no site oficial do WRF: `0` ou `1` para mosaico habilitado ou não; o caminho do papel de parede, que deve ser indicado usando duas barras invertidas (por este se tratar de um caractere de escape); e `0`, `1` ou `2` para papel de parede centralizado, lado a lado ou estendido. Ainda é importante ressaltar que todos esses valores devem estar separados por ponto e vírgula, como indicado no próprio programa. No caso deste exemplo, o mosaico está desabilitado, a imagem do papel de parede é ajustada para

```
1  REGEDIT4
2  % Generated with GWRF.
3
4
5  [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CDRom]
6  "Autorun"=dword:00000001
7  [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\
8  CurrentVersion\Winlogon]
9  "CachedLogonsCount"="0"
10 [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\
11 CurrentVersion\Winlogon]
12 "DeleteRoamingCache"=dword:00000001
13 [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
14 lanmanserver\parameters]
15 "srvcomment"="GWRF Test"
```

Figura 4.5: Exemplo de trecho de código gerado com o GWRF.

`C:\PearlJam.bmp` e o estilo fica definido como estendido. Além disso, a opção `Segurança` do menu de contexto do sistema é desabilitada, juntamente com a aba `Hardware` do item `Sistema`, no Painel de Controle. Neste caso, para as opções da figura 4.6, as linhas de código correspondentes, na figura 4.7, são, respectivamente, `Wallpaper image and style` (linhas 1 a 4), `No security tab` (linhas 5 a 7) e `No hardware tab` (linhas 8 a 10).

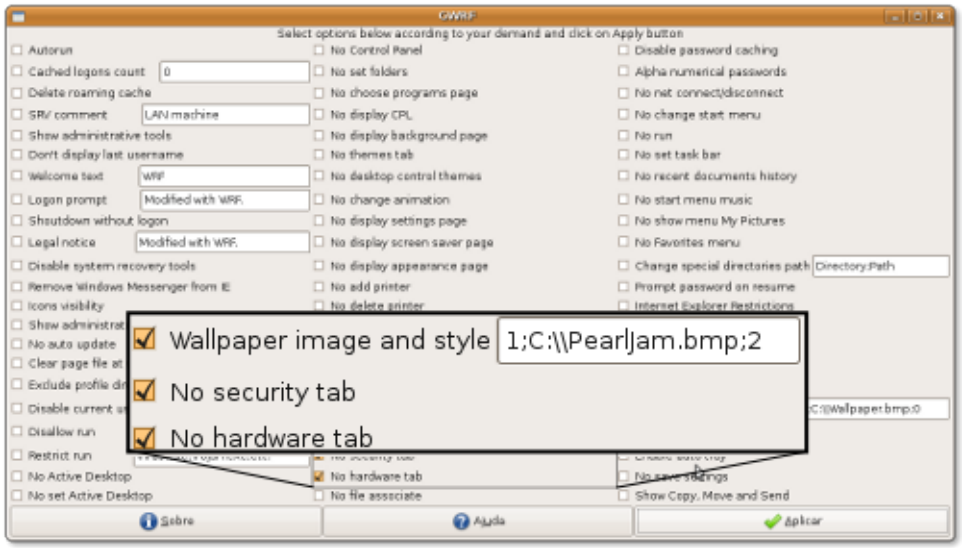

Figura 4.6: Segundo exemplo de configuração no GWRF.

```
1  [HKEY_CURRENT_USER\Control Panel\Desktop]
2  "TileWallpaper"="1"
3  "Wallpaper"="C:\\PearlJam.bmp"
4  "WallpaperStyle"="2"
5  [HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
6  Policies\Explorer]
7  "NoSecurityTab"=dword:00000001
8  [HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
9  Policies\Explorer]
10 "NoHardwareTab"=dword:00000001
```

Figura 4.7: Segundo exemplo de trecho de código gerado com o GWRF.

No último exemplo, apresentado pelas figuras 4.8 e 4.9, o papel de parede da tela de logon é alterado para `C:\Logon.bmp`, com mosaico desabilitado e estendido (assim como a configuração do papel de parede do sistema no segundo exemplo desta seção), o botão direito na barra de tarefas é desabilitado, os ícones inativos da bandeja do sistema são automaticamente escondidos, o salvamento de configurações do usuário é desabilitado e as opções `Copiar para`, `Mover para` e `Enviar para` são configuradas para serem exibidas no menu de contexto do sistema. Neste exemplo, a correspondência entre as figuras 4.8 e 4.9, são, respectivamente, `Logon wallpaper and style` (linhas 1 a 4), `No tray context menu` (linhas 5 a 7), `Enable auto tray` (linhas 8 a 10), `No save settings` (linhas 11 a 13) e `Show copy, move and send` (linhas 14 a 24).

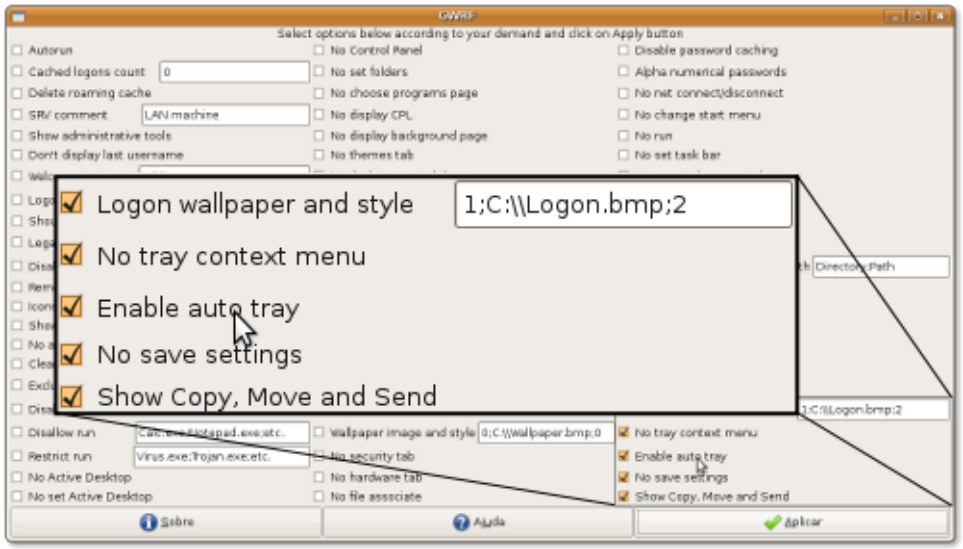

Figura 4.8: Terceiro exemplo de configuração no GWRF.

```
[HKEY_USERS\.DEFAULT\Control Panel\Desktop]
"TileWallpaper"="1"
"Wallpaper"="C:\\Logon.bmp"
"WallpaperStyle"="2"
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Policies\Explorer]
"NoTrayContextMenu"=dword:00000001
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Explorer]
"EnableAutoTray"=dword:00000001
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Policies\Explorer]
"NoSaveSettings"=dword:00000001
[HKEY_CLASSES_ROOT\AllFilesystemObjects\shellex\
ContextMenuHandlers\Copy To]
@="{C2FBB630-2971-11d1-A18C-00C04FD75D13}"

[HKEY_CLASSES_ROOT\AllFilesystemObjects\shellex\
ContextMenuHandlers\Move To]
@="{C2FBB631-2971-11d1-A18C-00C04FD75D13}"

[HKEY_CLASSES_ROOT\AllFilesystemObjects\shellex\
ContextMenuHandlers\Send To]
@="{7BA4C740-9E81-11CF-99D3-00AA004AE837}"
```

Figura 4.9: Terceiro exemplo de trecho de código gerado com o GWRF.

Capítulo 5

Considerações Finais

5.1 Conclusão

A utilização do WRF possibilita uma forma fácil de configuração de estações Windows dentro de um domínio. A centralização das contas de usuários permitiu que as mesmas fossem controladas automaticamente pelo PDC e que esse controle fosse garantido sempre que o usuário entrasse no domínio. Outra vantagem demonstrada por este procedimento foi a facilidade em se alterar as configurações de um grupo de usuários. Isto pois, para realizar esta tarefa, bastou gerar um novo arquivo de entrada para o Regedit e substituir o anterior no servidor.

A utilização do WRF e do GWRF facilitou muito a criação dos *scripts* de entrada para o Regedit. Este módulo escrito em linguagem Python cria uma camada de abstração para o administrador e isto possibilitou a confecção de *scripts* sem que fosse necessário conhecer quais chaves deveriam ser alteradas para se obter determinado resultado. Através dos testes realizados, o autor pode afirmar que o trabalho atendeu plenamente às expectativas iniciais, até mesmo superando-as. Com a criação do módulo WRF e do GWRF, a criação de *scripts* para o Regedit foi consideravelmente facilitada, tornando-se mais intuitiva, inclusive.

Atuando dentro do contexto deste trabalho, o WRF mostrou-se uma ferramenta bastante poderosa a favor do administrador. Com a alteração de um parâmetro de determinada função é possível criar um novo arquivo de script para o Regedit, que configura determinado aspecto do Windows e a simples substituição do antigo arquivo de configuração de grupo, por este, permite que vários usuários

do domínio tenham acesso a estas novas configurações, poupando tempo do administrador.

Pôde-se perceber, com a realização deste trabalho, a facilidade de se utilizar a linguagem Python. Foi possível para o autor, no prazo de aproximadamente um mês, aprender a linguagem e construir a primeira versão funcional do WRF. Esta facilidade, associada à legibilidade do código e à vasta API da linguagem propiciaram um ambiente bastante adequado para a construção da aplicação em questão, com um ambiente favorável a futuras melhorias no código e extensão do projeto.

Não deve ser muito difícil para administradores de servidores Samba implementarem a proposta deste trabalho, uma vez que, além das configurações do Samba, eles deveriam gerar o *script* para o Regedit, sendo que este trabalho é intermediado pelo WRF, que em conjunto com o GWRF, automatiza este processo.

5.2 Propostas para Trabalhos Futuros

Apesar do GWRF prover uma interface gráfica funcional para o WRF, automatizando a criação de *scripts* para o Regedit, o mesmo pode ser melhorado. As opções de criação de *scripts* poderiam estar melhor organizadas, talvez pela utilização de abas ou poder-se-ia utilizar outro *widget* para seleção das opções do usuário, diferente do *checkbox*, que é ostensivamente utilizado na interface. Outra possibilidade seria a criação de perfis de usuários na interface gráfica, ou seja, o administrador poderia criar, por exemplo, o perfil `aluno`, com todas as opções de criação do script já selecionadas. Então, caso futuramente ele precise gerar novamente ou alterar o *script* de configuração para os alunos, ele poderia selecionar o perfil que havia criado, alterá-lo e gerar novamente o *script* de configuração para os usuários. Além disso, a interface gráfica poderia permitir que o administrador selecionasse diretamente onde ele quer que os *scripts* gerados sejam gravados, incluindo seus nomes, seja na máquina local ou em uma máquina remota. Isto evitaria que ele tivesse que gerar o *script*, renomeá-lo e movê-lo para o diretório ou máquina correta. Neste contexto, poder-se-ia, ainda, criar uma interface para o WRF que funcionasse com passagem de parâmetros. Dessa forma seria possível criar *scripts* em Bash que interagissem com o WRF.

Fazer com que o GWRF pegue as opções de geração dos *scripts* a partir de um arquivo em XML seria outra proposta para melhoria do programa. Isto facilitaria

a adequação da interface a varias aplicações, além de tornar menos trabalhosa a inserção de mais opções.

Outro projeto interessante e, talvez, necessário, seja o de adaptação do WRF às novas versões do Windows, como o Vista e o 7. Certamente essas versões trarão novas opções de configuração e poderão alterar algumas antigas, tornando necessária a atualização do módulo gerador de *scripts*.

Esse projeto de atualização pode ser estendido para o Samba, inclusive. É provavel que futuras versões deste *software* possuam novas funcionalidades, como suporte ao Active Directory (AD) (serviço de diretório no protocolo LDAP em redes Windows). Com o suporte do Samba ao AD, o WRF poderia implementar uma forma de integração direta com o serviço de diretório da Microsoft, proporcionando uma gama maior de configurações que podem ser feitas e automatizadas através da edição direta do Registro do Windows, o que poderia facilitar ainda mais a realização de tarefas administrativas.

Pensando nos usuários de Windows, poder-se-ia criar uma forma de adaptar o GWRF para abertura dentro desse sistema, pois assim eles poderiam configurá-lo de acordo com as suas necessidades, fora do ambiente de rede.

Referências Bibliográficas

BRUECK, D.; TANNER, S. *Python 2.1 Bible*. 1. ed. New York: Wiley, 2001.

ECKSTEIN, R.; COLLIER-BROWN, D.; KELLY, P. *Using Samba*. 1. ed. Cambridge: O'Reilly, 1999. Acessado em 4 de julho de 2008. Disponível em: <http://www.unix.com.ua/orelly/samba/>.

FINLAY, J. *PyGTK 2.0 Tutorial*. New York, 2005. Acessado em 7 de junho de 2009. Disponível em: <http://www.pygtk.org/pygtk2tutorial/index% -.html>.

HAYDAY, J. *Segurança para Microsoft Windows 2000*. 1. ed. Rio de Janeiro: Campus, 2001.

HERTEL, C. Samba: An Introduction. *Samba.org*, 2001. Acessado em 30 de junho de 2008. Disponível em: <http://us3.samba.org/samba/docs/SambaIntro.html>.

HONEYCUTT, J. *Microsoft Windows Registry Guide*. 2. ed. Redmond: Microsoft Press, 2005.

Microsoft Corporation. *Configuração do Windows XP: Conecte-se à Rede*. Redmond, 2005. Acessado em 27 de julho de 2008. Disponível em: <http://www.microsoft.com/brasil/windowsxp% -/using/setup/getstarted-/configconnect.mspx>.

NBSO. *Práticas de Segurança para Administradores de Redes Internet*. Rio de Janeiro, 2003. Acessado em 28 de junho de 2008. Disponível em: <http://www.cert.br/docs/seg-adm-redes/seg-adm-redes.html>.

NEMETH, E.; SNYDER, G.; HEIN, T. R. *Manual Completo do Linux: Guia do Administrador*. 2. ed. São Paulo: Pearson, 2007.

Net Applications. *Operating System Market Share*. 2009. Internet. Acessado em 24 de agosto de 2009. Disponível em: <http://marketshare.hitslink.com-/operating-system-market-share.aspx?qprid=8>.

Python Software Foundation. *General Python FAQ*. Hampton, 2008. Acessado em 19 de julho de 2008. Disponível em: <http://www.python.org/doc/faq/general/>.

Samba Team. *smb.conf Man Page*. Lochiel, 2008.

SILVA, G. M. d. *Guia Foca GNU/Linux*. Vila Velha, 2007. Acessado em 15 de julho de 2008. Disponível em: <http://focalinux.cipsga.org.br/guia/avancado/ch-s-samba.html>.

SILVA, J. M. d. Uso do Samba como PDC em uma rede mista criando políticas de uso e autenticação. 2004. Monografia aprovada pelo curso de Administração de Redes Linux (ARL).

Unicode Consortium. *The Unicode Consortium*. 2009. Internet. Acessado em 7 de abril de 2009. Disponível em: <http://www.unicode.org/standard-/WhatIsUnicode.html>.

VENNERS, B. *The Making of Python*. 2003. Internet. Acessado em 27 de julho de 2008. Disponível em: <http://www.artima.com/intv/pythonP.html>.

Apêndice A

Tipos de Valores do Registro do Windows

Nome	Descrição
REG_BINARY	Dado binário, manipulado através de notação hexadecimal pelo Regedit, e.g., `0x02 0xFE 0xA9`.
REG_DWORD	São comumente usados para armazenar valores booleanos, com `0` indicando falso e `1`, verdadeiro. Podem ser editados em notação decimal ou hexadecimal, e.g., `0x00000001`.
REG_DWORD_BIG_ENDIAN	Valores DWORD com os bytes mais significantes armazenados primeiro na memória, e.g., o valor `0x01020304` é armazenado na memória como `0x01 0x02 0x03 0x04`.
REG_DWORD_LITTLE_ENDIAN	Teoricamente é o inverso do BIG_ENDIAN, mas assemelha-se mais ao *DWORD*, tanto que o Regedit não dá a opção para criação deste tipo, e.g., o valor `0x01020304` é armazenado na memória como `0x04 0x03 0x02 0x01`.
REG_EXPAND_SZ	Texto de tamanho variável. É usado normalmente para armazenar variáveis de ambiente.
REG_FULL_RESOURCE_DESCRIPTOR	Usado em dispositivos pelo SO e o administrador não tem muito controle sobre este tipo, tanto que o Regedit não permite sua manipulação.
REG_LINK	*Link*. Não pode ser criado pelo administrador.
REG_MULTI_SZ	Valores binários que contém listas de *strings*. Cada caractere deste tipo é separado do outro por um caractere nulo (`0x00`) e é terminado com dois caracteres nulos (Regedit versão 5).
REG_NONE	Valores sem tipos definidos.

Nome	Descrição
REG_QWORD	Valores *Quadruple-word* de 64 bits. Funciona como um *DWORD*, mas com o dobro de bits (presente apenas no Windows XP Pro de 64 bits).
REG_QWORD_BIG_ENDIAN	Funciona de forma análoga ao REG_DWORD_BIG_ENDIAN, mas com *QWORD* de 64 bits.
REG_QWORD_LITTLE_ENDIAN	Funciona de forma análoga ao REG_DWORD_LITTLE_ENDIAN, mas com *QWORD* de 64 bits.
REG_RESOURCE_LIST	Não é nada além de uma lista de valores REG_FULL_RESOURCE_DESCRIPTOR que o Regedit permite apenas ao administrador visualizar.
REG_RESOURCE_REQUIREMENTS_LIST	Lista de recursos que os dispositivos requerem. Assim como no anterior, o administrador pode apenas visualizar.
REG_SZ	Texto de tamanho específico. Termina com um caractere nulo.