CENTRO UNIVERSITÁRIO ESTÁCIO JUIZ DE FORA
ESPECIALIZAÇÃO EM ADMINISTRAÇÃO E SEGURANÇA
DE SISTEMAS COMPUTACIONAIS

INCOMPATIBILIDADES DA MAQUINA VIRTUAL JAVA E PROTOCOLO SAMBA NO AMBIENTE UNIX-LIKE

Juiz de Fora
2017

HENRIQUE RENALDO DE MELO

INCOMPATIBILIDADES DA MAQUINA VIRTUAL JAVA E PROTOCOLO SAMBA NO AMBIENTE UNIX-LIKE

Projeto de Pesquisa apresentado ao Curso de Pós Graduação em Administração e Segurança de Sistemas Computacionais do Centro Universitário Estácio Juiz de Fora como requisito parcial para obtenção do título de pós-graduado em Administração e Segurança de Sistemas Computacionais.

Juiz de Fora
2017

HENRIQUE RENALDO DE MELO

INCOMPATIBILIDADES DA MAQUINA VIRTUAL JAVA E PROTOCOLO SAMBA NO AMBIENTE UNIX-LIKE

Projeto de Pesquisa apresentado ao Curso de Pós Graduação em Administração e Segurança de Sistemas Computacionais do Centro Universitário Estácio Juiz de Fora como requisito parcial para obtenção do título de pós-graduado em Administração e Segurança de Sistemas Computacionais.

Faculdade Estácio de Sá de Juiz de Fora
Professor Esp. Anderson Vieira (orientador)

INCOMPATIBILIDADES DA MAQUINA VIRTUAL JAVA
E PROTOCOLO SAMBA NO AMBIENTE UNIX-LIKE

RESUMO

Este artigo expõe por meio de experimentos a busca para o entendimento sobre o comportamento JVM no uso do protocolo SAMBA-CIFS. Entendendo que tal análise se faz necessária no aspecto de segurança em ambos os objetos de pesquisa. Porém nossa pesquisa se baseia em termos teóricos experimentais. Propondo uma contribuição para um caso particular de uso de aplicações Java nos ambiente UNIX-LIKE.

Palavras-chave: JVM. Java. APPLET. SAMBA. CIFS. APPLET. Protocolo.

ABSTRACT

This papers exposes the JVM behavior in the use of the SAMBA-CIFS protocol. Understanding that such analysis is necessary in the security aspect of both search objects. However our research is based on experimental theoretical terms. Applications based a Java, for a particular case of using Java applications in the UNIX-LIKE environment.

Key words: JVM. Java. APPLET. SAMBA. CIFS. APPLET. Protocol.

LISTA DE ABREVIATURAS E SIGLAS

AD – Active Directory

CIFS – Common Internet File System

DHCP – Dynamic Host Configuration Protocol

DNS – Domain Name System

IP – Internet Protocol

LDAP – Lightweight Directory Access Protocol

NetBIOS – Network Basic Input Output System

PDC – Primary Domain Controller

SMB – Server Message Block

SUMARIO

1. INTRODUÇÃO

O uso de recursos compartilhados é comum e funcional para ambientes de produção com computadores interligados em rede, consiste em um gerenciamento de contas de usuário, arquivos e impressão. A pesquisa experimental proposta busca entender o comportamento de recursos computacionais *Java Virtual Machine* (JVM) no uso do protocolo SAMBA-CIFS. Tais questões serão abordadas de forma teórica e experimentais buscando uma solução ideal hipotética; ainda que parte da mesma possa ser alcançada, alguns problemas serão explicitados. Um estudo de caso em ambiente UNIX-LIKE demonstrou que embora a JVM continue apresentando bons resultados quanto a sua proposta, observa-se que a mesma não possui acesso direto a navegação em rede nos protocolos SAMBA-CIFS. As aplicações que fazem uso desta tecnologia para navegar diretamente na rede, sem mapeamento, não conseguem. Por exemplo: Eclipse, Netbeans e Applet na Integração Bancária oferecido via plataforma web por Banco do Brasil. Acredita-se que isto seja uma política interna de segurança da JVM. Assim tal situação dificulta o uso de aplicações que utilizam JVM atualmente fornecido pela empresa Oracle. A tarefa da *Java Virtual Machine* é converter os *bytecodes* Java para as instruções nativas que o processador é capaz de executar, (LINDHOLM e YELLIN., 2014, p 02).

> […] The Java Virtual Machine knows nothing of the Java programming language, only of a particular binary format, the class file format. A class file contains Java Virtual Machine instructions (or bytecodes) and a symbol table, as well as other ancillary information. [1]

Os *bytecodes* torna as aplicações Java capazes de serem executadas em qualquer tipo de hardware, desde que exista uma implementação da JVM para a arquitetura desejada. Atualmente tal plataforma tem alcançado inúmeros dispositivos, desde computadores, robôs, celulares, pequenos dispositivos eletrônicos e a promissora linha *smart* de eletrônicos (smartphone, smarttv, smartwatch...). Todavia a base teórica experimental deste trabalho foi desenvolvida através de dispositivos *Desktops*, arquitetura 32bits/64bits, processadores Intel/AMD.

[1] A máquina virtual de Java não faz conhecimento da linguagem de programação de Java, somente de um formato binário particular, o formato da lima da classe. Um arquivo de classe contém instruções Java Machine Virtual (ou bytecodes) e uma tabela de símbolos, bem como outras informações auxiliares.

Por meio de experimentos fez-se a busca ao entendimento de tal comportamento na JVM e incluir meios para que o acesso da JVM a navegação de protocolos SAMBA-CIFS produza resultados satisfatórios. Embora os recursos de navegação estejam disponíveis nativamente no ambiente UNIX-LIKE as aplicações dependem que estes estejam disponibilizados na JVM; assim os meios a serem explorados estarão explícitos neste artigo. Tal tema está sendo abordado para mitigar problemas de uso dos recursos computacionais utilizando particularmente JVM e SAMBA no ambiente UNIX-LIKE. ambiente experimentados se basearam em plataformas Debian[2]/Ubuntu[3]. Hipoteticamente, com base em pesquisas tal problema se estende para os sistemas operacionais UNIX-LIKE. Tal hipótese não foi confirmada visto a enorme diversidade ofertados atualmente. Buscando então focar em amostras experimentais e base de pesquisa dos recursos envolvidos.

2. ANÁLISE JVM

Segundo (PRESSMAN, 2006), as mudanças ocasionadas devido as novas regras e pontos de segurança, contribuem para que softwares caminhem para uma melhoria constante. Mudanças frequentes se fazem durante sua linha de produção. No seguimento de produção da plataforma Java, diversas soluções tecnológicas compõe a mesma. Possibilitando sua adequação aos dispositivos mais conhecidos atualmente no mercado. Todavia o Java é uma especificação bem definida por seus idealizadores. Contribuições ao longo do tempo tem amadurecido a proposta e disponibilizado recursos e melhorias de segurança. Assim vários produtores desta tecnologia disponibilizam as mais diversas ramificações da mesma, com particularidades especiais. Algumas distribuições Java saem um pouco à filosofia inicial, com acesso direto ao dispositivo: Ao exemplo de algumas distribuições Java para *smartphones*. Todavia focar-se-á na especificação e distribuição padrão, oriunda inicialmente da Sun Microsystems; e após sua aquisição pela Oracle Corporation em 2010. Mudanças e melhorias contribuem de certa forma para a continuidade desta filosofia, refinando regras e adequando a novas plataformas são oferecidas como soluções aos mais diversos tipos de dispositivos.

[2] Debian: Sistema Operacional UNIX-LIKE, disponível em: <https://www.debian.org>. Acesso em 1 de setembro de 2016.
[3] Ubuntu: Sistema Operacional UNIX-LIKE, disponível em: <http://www.ubuntu.com>. Acesso em 1 de setembro de 2016.

O Java possui uma class library (conjunto padrão de classes que suportam a plataforma) específica (LINDHOLM & YELLIN, 1999). Desta forma, para que um aplicativo Java seja executado em determinada plataforma, estas instruções são traduzidas por uma Java Virtual Machine para instruções de baixo nível específicos para a plataforma de hardware utilizada. Tornando assim os aplicativos Java portáveis. Atualmente a plataforma é composta por alguns seguintes segmentos, cada um específico para sua aplicação. Segundo (LINDHOLM e YELLIN., 2014, p. 01) a JVM é a pedra angular da plataforma Java:

> […] The Java Viurtual Machine is the conerstone of the Java platform. It is the component of the technology responsible for its hardware and operationg system independence, the small size of its compiled code, and its ability to protec users form malicious programs.
> The Java Virtual Machine is an abstract computing machine. Like a real computing machine, it has an instructions set and manipulates various memory areas at run time. [4]

O Java SE (*Standard Edition*), suporta basicamente as aplicações Java Desktop. Inclui na sua class library APIs para comunicação via rede, tratamento de entrada e saída, construção de ambientes gráficos, transformação e manipulação de dados, funções matemáticas, suporte a internacionalização de aplicações, entre outras APIs mais específicas.

O Java EE (*Enterprise Edition*), suporta aplicações Web. Este segmento foi o que popularizou a plataforma Java, devido à alta padronização de código, padrões de projetos (*design patterns*), robustez e suporte a muitos servidores diferentes, por ser independente de hardware. Inclui as interfaces e especificações dos core patterns para Web, suporte a servlets (classes executadas diretamente por um servidor Web), suporte a JSP (um padrão de linguagem de script voltado para a Web), suporte a containers JEE (servidores de aplicação que oferecem recursos como transações, persistência de dados, controle de sessões e outros).

---

[4] A máquina Java Virtual é a pedra angular da plataforma Java. É o componente da tecnologia responsável por sua independência do hardware e do sistema operacional, o tamanho pequeno de seu código compilado e sua capacidade de proteger os usuários de programas maliciosos.
A Máquina Virtual Java é uma máquina de computação abstrata. Como uma máquina de computação real, ele tem um conjunto de instruções e manipula várias áreas de memória em tempo de execução.

Java ME (*Micro Edition*) – Segmento voltado para dispositivos móveis e embarcados como celulares, PDAs e outros. Possui suporte a apenas um subset do Java SE, permitindo que as aplicações deste segmento sejam executadas em dispositivos com recursos (processamento, interface e memória) limitados. As aplicações deste segmento são usualmente chamadas de midlets.

O Java Card, segmento voltado para execução de aplicações Java em *Smart Cards*. Um típico exemplo de aplicação de Java Card é o SIM chip, utilizado em celulares GSM para autenticação e execução de serviços GSM. Um SIM chip consiste de um computador completo, com um processador, interface de entrada e saída, memória RAM e ROM. O Java Card foi desenvolvido essencialmente para este tipo de hardware reduzido, sem nenhuma interface com o usuário. JavaFX – Segmento voltado para a execução de sistemas multimídia na Web em Desktop e celulares. Acredita-se que este segmento seja melhor aproveitado em aplicações ricas para a Web.

O desenvolvimento de aplicações para toda a plataforma Java é suportado basicamente por um kit de desenvolvimento chamado de JDK (*Java Development Kit*). Este kit possui um compilador Java, ferramentas para a linguagem Java, ferramentas de debug e testes. Um código Java, depois de compilado, se transforma numa sequência de *bytecodes* similares aos microcódigos de instruções utilizados pelos processadores. Porém, diferentemente de outros microcódigos, os bytecodes Java não são otimizados para processamento em nenhum tipo de máquina. Os *bytecodes* Java são instruções genéricas, que são comumente implementadas de formas diferentes em hardwares diferentes, mas que possuem um mesmo objetivo. Através dos *bytecodes* Java, pode-se construir uma arquitetura virtual que fosse facilmente adaptada a qualquer tipo de máquina através de um ambiente de execução de código Java. Este ambiente é chamado de JRE (*Java Runtime Environment*) e é composto de uma máquina virtual (*Java Virtual Machine*) mais as classes de aplicação do segmento para qual a aplicação foi construída.

3. ANÁLISE SAMBA-CIFS

Protocolos de redes são regras usadas para que haja a comunicação de dados entre computadores. Eles fazem com que dois ou mais computadores possam trocar mensagens entre si. (COMER, 2007). Os protocolos(HERTEL, 2003) envolvidos

para implementação do Samba 4 AD são: DNS, LDAP, KERBEROS, NTP, SMB/CIFS, DHCP. Segundo Tanenbaum e Wetherall (2011), em uma rede de computadores a comunicação entre as máquinas é feita através do endereçamento IP. Alguns programas que empregam o endereço IP (KUROSE, 2010) podem ser concebidos por páginas de internet, correios eletrônicos, serviços de rede local, entre outros serviços. Por existirem inúmeras combinações de endereços IPs, torna-se impossível para as pessoas gravarem os mesmos. Surgiu assim a necessidade do desenvolvimento de um mecanismo para converter os IPs em nome e vice-versa. Para facilitar criou-se o DNS. Segundo (COSTA, 2010, p.7), o TCP/IP ao ser implementado no SAMBA, trouxe benefícios, desde que esse esteja bem compreendido.

> [...] é importante entender como a identificação de máquinas funciona em uma rede SMB para facilitar a configuração do Samba.
> Uma rede TCP/IP identifica cada máquina com uma sequência de números, composto por quatro blocos com três dígitos cada, que podem ir de 0 a 255. Exemplo: 192.168.0.1

O protocolo CIFS evolui-se do SMB (COSTA, 2010) passando por várias melhorias. O Samba nada mais é do que uma implementação das mesmas funções para sistemas do mundo UNIX. O seu desenvolvimento foi baseado no SMB, e assim, foi evoluindo e atualizando-se, introduzindo suporte ao CIFS. Mantendo-se atualizado lado a lado com as versões mais recentes do Windows. (MORIMOTO, 2009).

O servidor Samba possibilita o compartilhamento de arquivos e impressoras, também atua como um PDC, autenticando os usuários em redes locais(TERPSTRA, 2005). O Samba é um serviço de rede bem completo e flexível, atende muito bem clientes Linux, quanto clientes Windows. (MORIMOTO, 2009). Samba é composto por dois programas-chave, que são smbd e o nmbd. Suas funções de trabalho são a implementação dos quatro serviços básicos CIFS, que são: serviços de arquivo e impressão, autorização e autenticação, resolução de nomes e por último, anúncio de serviço de navegação. (SAMBA.ORG, 2015).

Os serviços de arquivo e impressão são o carro chefe da suíte CIFS. Estes são fornecidos pelo smbd, o SMB Daemon. Smbd também lida com o modo de autenticação e autorização. Já as outras duas peças CIFS, resolução de nomes e de navegação, são tratados pelo nmbd. Estes dois serviços envolvem basicamente a

gestão e distribuição de listas de nomes NetBIOS. A resolução de nomes assume duas formas: de broadcast e ponto-a-ponto. A máquina pode utilizar qualquer um ou ambos estes métodos, dependendo da sua configuração. (SAMBA.ORG, 2015).

O compartilhamento de recursos abrange dispositivos, arquivos e pastas. E há diferença quanto a isto também em cada Sistema Operacional. Visto que a plataforma UNIX-LIKE enxerga tudo como arquivo. Qualquer recurso nela é representado como arquivo; sendo um Sistema Operacional orientado a arquivo. Assim qualquer recurso pode ser disponibilizado via protocolo SMB. Este funciona como uma cada realizando o suporte entre as partes que se relacionam no trabalho realizado. O Dr. Barry Feigenbaum, funcionário da IBM na década de 90, implementou o protocolo inicialmente para funcionamento sobre a API NetBIOS/NetBEUI. A parti do Windows 2000 o mesmo já trabalha com o protocolo TCP (SHARPE, 2002). Tal protocolo é comumente usado no compartilhamento de objetos em uma rede ou domínio. Sua disponibilizada se restringe a regras internas de configuração e políticas de acesso a rede onde a mesma está participando.

O serviço provido via protocolo SMB disponibiliza de forma transparente para os hosts envolvidos o domínio-alvo solicitado. Suas requisições se fazem, por exemplo, ao ler, salvar, abrir ou fechar um objeto. Sua autenticidade é checada por e disponibilizada assim que a confirmação se faz entre as partes cliente-servidor; Liberando ou negando o recurso conforme permissão gerenciado via protocolo (SHARPE, 2002). A razão principal que fez o protocolo SMB receber contínuas modificações foi a falta de suporte a autenticação. Qualquer usuário poderia escrever e ler tudo que estava a seu alcance, o que não era muito adequado para grandes empresas. A necessidade de separação e estabelecer privilégio aumentaram então mais tarde a Microsoft estendeu o protocolo SMB para incluir suporte a gerenciamento de rede e outros serviços. O protocolo SMB/CIFS possui diversas funções associadas a ele (COSTA, 2010, p 7).

> [...] O protocolo SMB/CIFS funciona enviando pacotes de uma máquina para outra [...] A máquina que recebeu a requisição verifica, então, se a mesma é válida[...] funções associadas a ele. É possível compartilhar impressoras, definir níveis de segurança e autenticação, etc.

O protocolo que originalmente era apenas para compartilhamento incorporou outras funcionalidades e foi renomeado para CIFS; este é uma melhoria do protocolo

SMB. Implementado pela Microsoft atingindo uma parceria de disponibilização de recursos com outras plataformas, ou seja, desenvolvido para trabalhar independentemente da arquitetura de rede, protocolo de transporte e sistema operacional. Independência tal alcançada via RPC (Remote Procedure Call). Onde seus protocolos se baseiam no XDR (External Data Representation) (UFRJ, 2009).

A biblioteca RPC permite que o usuário – cliente venha utilizar uma aplicação executando uma chamada a outro usuário – servidor – de forma que pareça que o mesmo esteja em seu próprio espaço (UFRJ, 2009). A XDR é um protocolo para apresentar dados. Realiza transferência de informações entre máquinas de arquiteturas e sistemas operacionais diferentes. Desta forma é utilizado pelo RPC para resolver problemas de compatibilidade entre máquinas e sua forma de interação (UFRJ, 2009).

No CIFS os pacotes são transmitidos entre os hosts envolvidos, onde cada pacote transmitido inclui algum tipo de requisição. O computador que recebeu a requisição verifica se tal é válido e se o recurso está disponível. Sendo a resposta positiva, então a mesma é informada ao host que a requisitou. O CIFS inicialmente foi disponibilizado para compartilhamento de objetos, sendo ele arquivos, impressoras. Devido a sua aceitação no mercado comparativo, demais recursos foram disponibilizados , como autenticação, níveis de segurança, e demais função via PDF. Segundo Diógenes e Mauser(2013, p. 3), segurança de redes de computadores possui três pilares fundamentais:

> [...] **Confidencialidade:** Trata-se da prevenção do vazamento de informação para usuários ou sistemas que não estão autorizados a ter acesso a tal informação.
> **Integridade:** Trata-se da preservação/manutenção do dado na sua forma íntegra, ou seja, sem sofrer modificações através de fontes não autorizadas.
> **Disponibilidade:** Trata-se da manutenção da disponibilidade da informação, ou seja, a informação precisa estar disponível quando se necessita.

Os compartilhamentos existentes no servidor sofrem restrições de acesso e são liberadas depois que o usuário se autentique com login e senha. Para a parte de autenticação a senha é enviada de forma criptografada. Diferente do protocolo SMB, o protocolo CIFS suporta múltiplas requisições, isso é feito através do uso de um id multiplexada (MID). Cada uma das requisições enviadas ao servidor possui um único MID e essas requisições são respondidas com o mesmo MID. Assim, múltiplas requisições podem ser enviadas ao servidor e quando o cliente receber a resposta

precisará comparar os MID para saber qual requisição foi respondida. O protocolo CIFS é uma extensão melhorada do protocolo SMB e disponibiliza acesso transparente de arquivos compartilhados em rede, foi criado pela Microsoft para trabalhar em qualquer tipo de plataforma.

4. EXPERIMENTOS

Num cenário onde as aplicações Java necessitam acessar recursos que estão na rede fornecida pelo protocolo SAMBA, tal recurso encontrava-se indisponível. Logo a JVM se restringe a acessar recursos locais. Tal cenário foi elaborado com inúmeros testes durante quase dois anos; os recursos mais explorados foram ferramentas de desenvolvimento de Software(Eclipse, Netbeans) e integração Bancária(applet bb.com.br) onde necessitava ler arquivos remotamente para troca de informações. Todavia tais condições se mostrarem pertinentes e inacessíveis de execução, salvo o fato de simular um recurso de acesso indireto que tem atingido com certo êxito, visto que o tal cenário é um ambiente controlado. Onde auditorias constantes sobre o seu funcionamento tem sido observadas de forma satisfatória.

Observado a restrição da JVM a recursos locais; partiu-se do princípio que se fosse possivel simular o acesso a conteúdo remoto de forma localmente, poder-se-ia obter o resultado esperado. Assim sendo as pesquisas de protocolos obteve tais recursos. Contudo o SAMBA não tinha nativamente suporte para isto. Não obteve sucesso nas pesquisas em repositórios definidos pelos fornecedores dos sistemas operacionais que estavam sendo usaros no experimento, a saber: Debian8/Ubuntu14/15. Apesar disto após análise em documentos disponíveis publicamente, principalmente em SAMBA.ORG. Consegui-se o resultado esperado. Ainda que alguns problemas de contexto envolvendo caracteres especiais em nomes dos arquivos apareceram.

Simulando um acesso localmente aos arquivos, que neste caso são os objetos indisponíveis na rede SAMBA remotamente para a JVM; conseguiu-se obter acesso ao recurso que antes estava somente disponível via navegação na rede SAMBA. No entanto tal simulação se deu em forma de “mapear” o domínio-objeto para a maquina local usando o protocolo MOUNT-CIFS.

Após uso por aproximadamente dois anos, observou-se o funcionamento adequado do mesmo; porém problemas na leitura no nome dos arquivos anda

continuavam a aparecer, mesmo que em caráter experimental. Visto que no ambiente de produção este fato não era apresentado, já que não havia a necessidade de trabalhar com caracteres especiais em nomes de arquivos. Quando os nomes dos arquivos possuem caracteres especiais, tais objetos são impedidos de renomear ou ser excluídos. Assim somente excluindo/renomeado o mesmo diretamente no servidor seria possível continuar. Contudo nesta empresa para continuidade de trabalhos com aplicações JAVA não se fez necessário tal ação; já que neste cenário tal anomalia não era reproduzida em ambiente de produção; por ausência dos caracteres especiais na situação observada.

Aplicado o protocolo, houve a necessidade de autenticação do devido usuário as suas permissões via protocolo SAMBA no domínio-alvo solicitado. Na hipótese de bibliotecas de registros de senhas existentes, seria solicitada a autenticação, o que não foi obtido êxito. Partindo então para solicitação dos dados de autenticação no início da sessão do usuário, conseguiu-se um mapeamento automático com as devidas permissões e simulando um "mapeamento" local de dados remotos.

Assim chegou-se a bons resultados com a experimentação, onde as aplicações Java via JVM conseguiram acesso ao objetos de forma local, ainda que o mesmo se encontrava remotamente. Todavia este recurso atualmente(2017) não se encontra disponível nativamente nas plataformas UNX-LIKE. Dependência/pacotes utilizados em plataforma Debian/Ubuntu: cifs-utils. Ferramenta de apoio smbmap, desenvolvida para “mapeamento”:

```
#!/bin/bash
# Source: github.com/henrymelo/smbmap
# Path script: /usr/bin/smbmap
# Dependencias: apt-get install cifs-utils
#
# All users sudo, add line end of file /etc/sudoers
# %sudo ALL=(root:ALL) NOPASSWD: /sbin/mount.cifs
# Or:
# Only usuariox, add line end of file /etc/sudoers
# usuariox ALL=(root:ALL) NOPASSWD: /sbin/mount.cifs
#
# For ~/.smbmap/smb.map, netPath:alias, example:
# //192.168.0.8/workspace:workspace
# //192.168.0.8/ambtstb:ambtstb
#
# For auto start by user, add line end of file /etc/profile
# smbmap

checkskel() {
  if [ ! -e /var/log/samba/log.smbmap ] ; then
    echo "Please: sudo touch /var/log/samba/log.smbmap"
    echo "Please: sudo chmod 777 /var/log/samba"
    return 1
  fi
```

```
   echo "" >> /var/log/samba/log.smbmap 2>&1
   DATETIME=`date`
   echo $DATETIME >> /var/log/samba/log.smbmap 2>&1

   if [ ! -d ~/.smbmap ] ; then
      mkdir ~/.smbmap
   fi
   if [ ! -e ~/.smbmap/smb.map ] ; then
      touch ~/.smbmap/smb.map
      echo "# Please, add line end of file: netPath:alias, example:" > ~/.smbmap/smb.map
      echo "# //192.168.0.8/workspace:workspace" >> ~/.smbmap/smb.map
      echo "# //192.168.0.8/ambtstb:ambtstb" >> ~/.smbmap/smb.map
   fi

   if [ ! -e /media/$USER/smbmap ] ; then
      echo "Please: sudo mkdir /media/$USER/smbmap" >> /var/log/samba/log.smbmap 2>&1
      echo "Please: sudo chmod 777 /media/$USER/smbmap" >> /var/log/samba/log.smbmap 2>&1
      return 2
   fi

}

mountsmb() {
   UIDx=`id -u $USER`
   GIDx=`id -g $USER`
   fARRMAP=`grep -v '#' ~/.smbmap/smb.map`
   ARRMAP=($fARRMAP)
   if [ ! "${ARRMAP[1]#*:}" ] ; then
      echo "Please, add line end of file ~/.smbmap/smb.map: netPath:alias, example:"
      echo "//192.168.0.8/workspace:workspace"
      echo "//192.168.0.8/ambtstb:ambtstb"
      return 3
   fi

   for MAP in "${ARRMAP[@]}" ; do
      SOURCE=${MAP%%:*}
      TARGET=${MAP#*:}
      mkdir /media/$USER/smbmap/$TARGET > /dev/null 2>&1

      ### begin “map” protocol MOUNT-CIFS ###
       sudo mount.cifs $SOURCE /media/$USER/smbmap/$TARGET cifs -o
           username=$USER,password=$SMBPASS,iocharset=utf8,gid=$GIDx,uid=$UIDx,file_mode=0777,
           dir_mode=0777 >> /var/log/samba/log.smbmap 2>&1 &
      ### end ###

   done
}

CHKGUI=`echo $TERM`
if [[ $CHKGUI != "dumb" ]] ; then
   exit 5
fi

checkskel > /dev/null 2>&1
SMBPASS=`zenity --password --title="Mapear pastas da rede" --add-password="smbpass" 2> /dev/null`
mountsmb >> /var/log/samba/log.smbmap 2>&1

# Return
case $? in
   1)
      echo "Please: sudo touch /var/log/samba/log.smbmap"
      echo "Please: sudo chmod 777 /var/log/samba"
   ;;
   2) echo "Please: sudo chmod 777 /media/$USER/smbmap";;
esac
```

A ferramenta de apoio ao “mapeamento” smbmap desenvolvida, se resume em “...sudo mount.cifs $SOURCE /media/$USER/smbmap/$TARGET cifs -o...”. Após conhecimento técnico sobre o protocolo, foi desejável automatizar o processo de forma menos custosa para o usuário. Envolvendo usabilidade e segurança para seus dados de autenticação. Liberando de forma adequada suas permissões para as necessidades que surgiam no uso das aplicações via JVM.

5. CONCLUSÃO

A filosofia Java é amplamente conhecida, sua popularização se deu por gerenciar recursos, liberando e proibindo recursos em tempo de execução para as aplicações Java. Estas produziam suas soluções com foco na JVM, e versão apropriada; sabendo que a mesma poderia ser executada em dispositivos heterogêneos. Contudo tais dispositivos diferenciam em disponibilidades de recursos nativos. A solicitação de tal condição em tese garantiria o mesmo; porém isto só pode ser assumido se o mesmo encontra-se disponível, senão tal condição não será verdadeira naquele momento para aquele dispositivo. Ainda que esteja disponível fisicamente, o software poderá conceder ou não negar acesso ao mesmo. Logo a proposta não seria alterar regras internas da JVM.

No entanto foi proposto a implementação de uma ferramenta de apoio às aplicações Java quando precisarem acessar a navegação de redes samba em ambiente UNIX-LIKE. Embora todos os recursos estivessem disponíveis para a JVM, a mesma não os explora ou compartilha entre suas aplicações, conforme foi o experimento citado. Atualmente há preocupações quanto a exploração de vulnerabilidades, porém não é o foco deste trabalho; no entanto isto explicaria a restrição de tal recurso.

Contribuições de um protótipo experimental da ferramenta desenvolvida com base nesta pesquisa; está funcionando há dois anos em uma empresa, com eficiência, já que a mesma simula um mapeamento local de um endereço de rede SAMBA-CIFS. Sua proposta não seria somente um script de apoio como atualmente está, mas uma integração mais adequada ao ambiente X do UNIX-LIKE, simulando “mapear unidade de rede”, conforme atualmente usado em outros sistemas operacionais.

6. REFERÊNCIAS BIBLIOGRÁFICAS

COMER, Douglas E., **Redes de computadores e internet**, 4. ed. Porto Alegre: Bookman. 2007.

COSTA, P. H. A., **Samba: Windows e Linux em rede**. São Paulo: Linux New Media do Brasil, 2010.

DIÓGENES, Yuri; MAUSER, Daniel, Certificação Security: Da prática para o exame SYO - 301, Rio de Janeiro: Novaterra, 2013.

HERTEL, Christopher, **Implementing CIFS: The Common Internet File System**, Upper Saddle River: Prentice Hall, 2003.

KUROSE, James F., ROSS, Keith W., **Rede de computadores e a Internet: uma abordagem top-down**, 5. ed. São Paulo: Addison Wesley, 2010.

MORIMOTO, Carlos E., **Servidores Linux**, Porto Alegre: Sul Editores, 2009.

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: McGraw-Hill, 2006.

TANENBAUM, Andrew S., WETHERALL, David, **Redes de Computadores**, São Paulo: Pearson Prentice Hall, 2011.

TERPSTRA, John. **Samba by Example**, Stoughton: Prentice Hall, 2005.

LINDHOLM, Tim, YELLIN. Frank, **The Java Virtual Machine Specification**, California, USA: Addson Wesley, 2014.

SAMBA-ORG. Disponível em <**https://www.samba.org/**>. Acessado em: 01 set. 2015