



## Vírus-scan: Solução para checagem de vírus

Data	Descrição	Autor
17/06/2013	Criação documento / solução	Denis Clayton Alves Ramos
01/07/2013	Apenas acréscimos de mais insumos do teste de performance	Denis Clayton Alves Ramos

## Vírus-scan: Solução para checagem de vírus

### Sumário

Vírus-scan: Solução para checagem de vírus.....	1
Introdução.....	2
Escopo.....	2
Premissas.....	2
Solução.....	2
Ponto de ATENÇÃO.....	4
Instalação.....	5
Conclusão.....	6
ANEXO 1 - configuração.....	7
ANEXO 2 – Resultado dos testes de desempenho.....	8
ANEXO 3 – Arquivo clamd.conf utilizado nos testes de performance.....	16
ANEXO 4 – Teste da instalação da servidora de aplicação + componente.....	28



# Vírus-scan: Solução para checagem de vírus

## Introdução

Este trabalho constituiu na pesquisa e criação de uma solução para interagir com programas anti-vírus. O objetivo é permitir que aplicações java acionem o anti-vírus para validar se arquivos carregados via sistema estão “limpos”.

## Escopo

Permitir que aplicações (online ou *batch*) possam escanear arquivos com motor de anti-vírus *ClamAV* instalado no servidor de aplicação.

## Premissas

- a) apresentar bom desempenho no processamento da checagem de vírus online, de forma a não impactar aplicações responsáveis por *upload* de documentos e fotos;
- b) ser adaptável para que mudanças da solução de anti-vírus (motor de anti-vírus) não impactem o código-fonte das aplicações clientes.

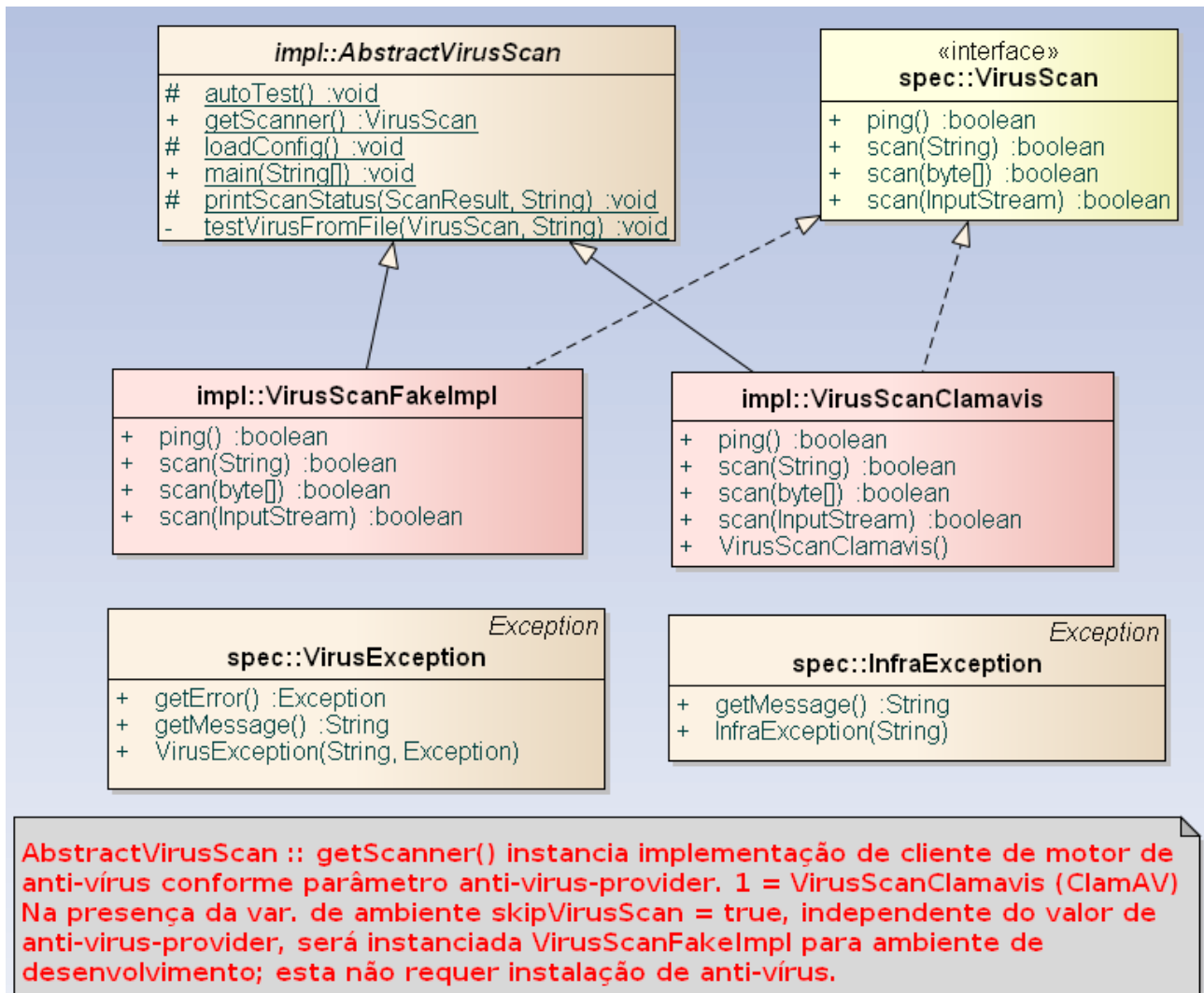
## Solução

Tendo em mente essas premissas, as seguintes tratativas foram tomadas:

- Em relação a premissa (a) pesquisando acerca do *ClamAV*, foi possível identificar uma forma de interação mais eficiente do que simplesmente encapsular uma chamada ao executável *clamscan*; solução mais comumente indicada em fóruns e onerosa. Assim, via *socket* são enviados comandos ao *daemon* do *ClamAV*. Além disso, o bom desempenho da checagem de vírus depende do *tunning* da solução de anti-vírus no servidor, como trabalhar de forma *multi-thread* e adequar parâmetros críticos como *MaxThreads*, *MaxQueue* etc.
- Em relação a premissa (b) a solução foi construída de maneira que ainda que se troque o motor de anti-vírus (e consequentemente a classe que implementa a comunicação com este), os clientes (aplicações) não sejam afetados.

Vide diagrama de classes abaixo:

## Vírus-scan: Solução para checagem de vírus



A partir da classe *AbstractVirusScan* (método *getScanner*) e do arquivo *virusScan-conf.properties* o cliente terá acesso (referência) a interface ***VirusScan*** que intermedia a implementação necessária conforme motor de anti-vírus em uso (*ClamAV* ou outro no futuro).



## Vírus-scan: Solução para checagem de vírus

Código-fonte requerido nas aplicações clientes:

```
VirusScan scanner = null;
try {
    scanner = AbstractVirusScan.getScanner();
    scanner.scan(foto.getConteudoArquivo());
    // lógica processamento arquivo limpo
} catch (VirusException e) {
    // lógica processamento arquivo infectado - recomenda-se rejeitar
    // upload. Exibir mensagem ao usuário final a partir de e.getMessage
    // ou adaptar para uma mensagem mais alto nível.
} catch (IOException e) {
    // Erro processamento arquivo. Exibir mensagem ao usuário final
    // para tentar novamente.
}
```

*foto.getConteudoArquivo()* deve ser substituído conforme arquivo a ser processado pela aplicação.

A classe *AbstractVirusScan* controla a solução – carrega arquivo de configuração, realiza auto-teste e instancia a implementação correta de acordo com anti-vírus *provider* configurado (motor).

E a interface *VirusScan* abstrai as funções de escanear arquivos.

O arquivo *virusScan-conf.properties* provê as propriedades **para** e **cc** usadas para enviar e-mail de alerta, caso o anti-vírus no servidor não esteja respondendo. A equipe responsável pelo sistema que usa a solução deve ser comunicada. Existem outras configurações neste arquivo (vide Anexo 1) .

### **Ponto de ATENÇÃO**

A API criada faz um auto-teste da solução, este é feito automaticamente via método *autoTest()* no carregamento da classe *AbstractVirusScan* pela JVM.

A cada 20 minutos, em uma *Thread* separada, é feito um teste de conexão com o anti-vírus no servidor, caso este não responda, um e-mail de alerta é encaminhado conforme propriedades do arquivo *virusScan-conf.properties*; além de exibir uma mensagem de erro no *log* da servidora de aplicação.

Assim, é importante configurar a propriedade **para** citada acima e opcionalmente a propriedade **cc**.



## Vírus-scan: Solução para checagem de vírus

### Instalação

Caso seu sistema não seja montado via *maven*: basta copiar **virus-scan-1.0.1.jar** para a pasta **WEB-INF/lib** de sua *webapp*.

Sistemas que são montados via *maven* devem proceder da seguinte forma:

- 1) No *pom.xml* raiz de seu sistema, adicione o repositório que contém a biblioteca virusScan “mavenizada” previamente:

```
<repository>
  <id>xx</id>
  <url>http://xx/maven2</url>
</repository>
```

- 2) Edite o *pom.xml* de sua *webapp* e adicione a seguinte dependência:

```
<dependency>
  <groupId>br.unicamp.ccuec.ctmq.components.virusscan</groupId>
  <artifactId>virus-scan</artifactId>
  <version>1.0.1</version>
</dependency>
```

- 3) Execute os comandos:

```
mvn eclipse:clean
```

```
mvn eclipse:eclipse
```

Ao obter sucesso na execução, faça um *refresh* do projeto (na sua IDE) para o qual adicionou a dependência no passo 2.

Após isto copie o código-fonte requerido do tópico “**Solução**” acima, adicione no código da aplicação e faça os ajustes comentados.

No próximo *build* que realizar deverá constar em **WEB-INF/lib** o **virus-scan-1.0.1.jar**.



## Vírus-scan: Solução para checagem de vírus

### 4) virusScan-conf.properties

É necessário copiar este arquivo para seu projeto *webapp* na IDE na pasta **src/main/resources**. Após montado o WAR, este arquivo deve constar em: **WEB-INF/classes**.

#### Observações:

1. Na servidora de aplicação deve estar instalada a solução de anti-vírus (motor) e as propriedades do arquivo *virusScan-conf.properties* devem ser adequadas conforme esta instalação;
2. Caso o desenvolvedor não queira instalar o anti-vírus em seu computador, basta setar o parâmetro **-DskipVirusScan=true** no *script* de *startup* do servidor de aplicação que a checagem de vírus será ignorada e o resultado dos métodos *scan* será *true* (sem vírus). Apenas o método *VirusScan :: scan(String)* que lançará *VirusException*, simulando assim arquivo com vírus para o desenvolvedor poder testar mensagem de erro para usuário.

## Conclusão

A solução apresentada visa cumprir ao máximo as premissas estabelecidas. Quanto a desempenho está evidenciado no tópico “**Anexo 2**” e quanto a adaptabilidade, está evidenciado no tópico “**Solução**” e “**Anexo 4**” a aderência a elas.



## Vírus-scan: Solução para checagem de vírus

### **ANEXO 1 - configuração**

Arquivo de configuração virusScan-conf.properties:

```
#propriedades para & cc sao os enderecos de e-mail a serem informados caso
#daemon anti-virus nao esteja respondendo ou executando
#usar caracter , como separador
para=clayton@ccuec.unicamp.br
#cc=
#host fortemente recomendado instalacao local
host=127.0.0.1
#porta padrao daemon
port=3310
#tempo maximo (milisegundos) para scan arquivo
timeout=120000
#anti-virus providers: ClamAV = 1 (default)
#anti-virus-provider=1
```



## Vírus-scan: Solução para checagem de vírus

### ANEXO 2 – Resultado dos testes de desempenho

Os testes de desempenho foram executados com o apoio da equipe Divisão de Produção e Suporte bem como o tuning do Anexo 3.

Aplicação: ficha inscrição estudante especial

Versões:

a) "Padrão" (siga4 head); versus

b) "Com checagem de vírus" (siga4 head mais chamada a componente de sw **virus-scan**)

Duração: 2h42

Ambiente: Homologação (2 servidoras na configuração vigente) com software *ClamAV* instalado.

Os números de concorrência exercitados nos testes abaixo, bem como o script *jmeter* usado nos testes foram obtidos da equipe que criou a aplicação em questão e fez essas estimativas.

#### A) Teste I: **20** usuários concorrentes (mesmo instante) com repetição de **30** vezes e **1** cliente *jmeter*

Neste teste cada página da aplicação é acionada 600 vezes, inclusive os uploads (foto / documento).

Versão aplicação	Transação (pág. web)	Tempo linha 90% (milisegundos)
Padrão	upload_candidato_foto [envio <b>foto</b> ]	4809
Com checagem de vírus	upload_candidato_foto [envio <b>foto</b> ]	4882
Padrão	upload_documentos [envio <b>arquivo</b> ]	9973
Com checagem de vírus	upload_documentos [envio <b>arquivo</b> ]	10067
Padrão	aba_endereco [busca por cep]	5660
Com checagem de vírus	aba_endereco [busca por cep]	6366 <sup>*1</sup>
Padrão	upload_documentos2 [envio <b>arquivo</b> ]	9813
Com checagem de vírus	upload_documentos2 [envio <b>arquivo</b> ]	9801 <sup>*2</sup>
Padrão	apresentar_ficha_inscricao_estudante_especial [botao salvar]	11759
Com checagem de vírus	apresentar_ficha_inscricao_estudante_especial [botao salvar]	11881





## Vírus-scan: Solução para checagem de vírus

Em 20130619-teste\_inscricao\_600\_antivirus.csv consta medidas completas de “Com Checagem de vírus”.

### Observações:

- 1) <sup>\*1</sup> busca por cep teve aumento 706 milisegundos.
- 2) <sup>\*2</sup> pequeno decréscimo do tempo.
- 3) Teste executado sem incidência de erros no processamento da aplicação.
- 4) Ao longo do teste: Programas batches executando em background no ambiente e acessei a mesma aplicação online **sem** percepção de lentidão no uso da mesma.

**B) Teste II: 20 usuários concorrentes (mesmo instante) com repetição de 50 vezes e 2 clientes jmeter**

Neste teste cada página da aplicação é acionada 2000 vezes, inclusive os *uploads* (foto / documento).

Versão aplicação	Transação (pág. web)	Tempo cliente1, cliente2 linha 90% (milisegundos)
Padrão	upload_candidato_foto [envio <b>foto</b> ]	7326, 7871
Com checagem de vírus	upload_candidato_foto [envio <b>foto</b> ]	7726, 7959
Padrão	upload_documentos [envio <b>arquivo</b> ]	12530, 13932
Com checagem de vírus	upload_documentos [envio <b>arquivo</b> ]	14533, 14737 <sup>*1</sup>
Padrão	aba_endereco [busca por cep]	11026, 11471
Com checagem de vírus	aba_endereco [busca por cep]	10316, 10678 <sup>*2</sup>
Padrão	upload_documentos2 [envio <b>arquivo</b> ]	12010, 13053
Com checagem de vírus	upload_documentos2 [envio <b>arquivo</b> ]	14078, 14280 <sup>*3</sup>
Padrão	apresentar_ficha_inscricao_estudante_especial [botão salvar]	18372, 18892
Com checagem de vírus	apresentar_ficha_inscricao_estudante_especial [botão salvar]	20414, 21255 <sup>*4</sup>



## Vírus-scan: Solução para checagem de vírus

Nos arquivos abaixo constam medidas completas de “Com Checagem de vírus”:

- 2013-06-19-teste\_inscricao\_1000\_antivirus\_cliente1.csv
- 2013-06-19-teste\_upload-1000-antivirus-cliente2.csv

### Observações:

- 1) \*<sup>1</sup> \*<sup>3</sup> \*<sup>4</sup> aproximadamente 2 segundos de acréscimo no tempo de resposta.
- 2) \*<sup>2</sup> pequeno **decréscimo** do tempo de resposta.
- 3) Teste executado sem incidência de erros no processamento da aplicação.
- 4) Ao longo do teste: Programas batches executando em background no ambiente e acessei a mesma aplicação online **com** percepção de lentidão no uso da mesma, mas nada que inviabilizasse seu uso.

### Conclusão:

Os tempos de resposta no teste I ficaram muito próximos entre as versões da aplicação, sem e com a checagem de vírus.

A carga do teste II representa mais que o dobro da carga do teste I. Nele houve degradação no tempo de resposta de 2 segundos. Este cenário de concorrência é agressivo, e o mesmo não foi vivenciado até mesmo no *Redefor* (8.000 candidatos) que também demandou *upload* de documentos e foto, porém sem a checagem do anti-vírus.

O benefício da checagem de vírus para um público externo à universidade (estudante especial) paga o custo à mais no processamento.

Importante: Arquivo *clamd.conf* da instalação do anti-vírus *ClamAV* das servidoras de homologação utilizadas nos testes de desempenho citados acima encontra-se no Anexo 3.



### ***ANEXO 3 – Arquivo clamd.conf utilizado nos testes de performance***

```
##  
## Example config file for the Clam AV daemon  
## Please read the clamd.conf(5) manual before editing this file.  
##  
  
# Comment or remove the line below.  
#Example  
  
# Uncomment this option to enable logging.  
# LogFile must be writable for the user running daemon.  
# A full path is required.  
# Default: disabled  
LogFile /var/log/clamav/clamd.log  
  
# By default the log file is locked for writing - the lock protects against  
# running clamd multiple times (if want to run another clamd, please  
# copy the configuration file, change the LogFile variable, and run  
# the daemon with --config-file option).  
# This option disables log file locking.  
# Default: no  
#LogFileUnlock yes  
  
# Maximum size of the log file.  
# Value of 0 disables the limit.  
# You may use 'M' or 'm' for megabytes (1M = 1m = 1048576 bytes)  
# and 'K' or 'k' for kilobytes (1K = 1k = 1024 bytes). To specify the size  
# in bytes just don't use modifiers.  
# Default: 1M  
LogFileMaxSize 0  
  
# Log time with each message.  
# Default: no  
LogTime yes  
  
# Also log clean files. Useful in debugging but drastically increases the  
# log size.  
# Default: no  
#LogClean yes
```



## Vírus-scan: Solução para checagem de vírus

```
# Use system logger (can work together with LogFile).
# Default: no
LogSyslog yes

# Specify the type of syslog messages - please refer to 'man syslog'
# for facility names.
# Default: LOG_LOCAL6
#LogFacility LOG_MAIL

# Enable verbose logging.
# Default: no
#LogVerbose yes

# Log additional information about the infected file, such as its
# size and hash, together with the virus name.
#ExtendedDetectionInfo yes

# This option allows you to save a process identifier of the listening
# daemon (main thread).
# Default: disabled
PidFile /var/run/clamav/clamd.pid

# Optional path to the global temporary directory.
# Default: system specific (usually /tmp or /var/tmp).
TemporaryDirectory /var/tmp

# Path to the database directory.
# Default: hardcoded (depends on installation options)
DatabaseDirectory /var/clamav

# Only load the official signatures published by the ClamAV project.
# Default: no
#OfficialDatabaseOnly no

# The daemon can work in local mode, network mode or both.
# Due to security reasons we recommend the local mode.

# Path to a local socket file the daemon will listen on.
# Default: disabled (must be specified by a user)
LocalSocket /var/run/clamav/clamd.sock

# Sets the group ownership on the unix socket.
# Default: disabled (the primary group of the user running clamd)
#LocalSocketGroup virusgroup
```



## Vírus-scan: Solução para checagem de vírus

```
# Sets the permissions on the unix socket to the specified mode.
# Default: disabled (socket is world accessible)
#LocalSocketMode 660

# Remove stale socket after unclean shutdown.
# Default: yes
FixStaleSocket yes

# TCP port address.
# Default: no
TCPsocket 3310

# TCP address.
# By default we bind to INADDR_ANY, probably not wise.
# Enable the following to provide some degree of protection
# from the outside world.
# Default: no
TCPAddr 127.0.0.1

# Maximum length the queue of pending connections may grow to.
# Default: 200
MaxConnectionQueueLength 30

# Clamd uses FTP-like protocol to receive data from remote clients.
# If you are using clamav-milter to balance load between remote clamd daemons
# on firewall servers you may need to tune the options below.

# Close the connection when the data size limit is exceeded.
# The value should match your MTA's limit for a maximum attachment size.
# Default: 25M
#StreamMaxLength 10M

# Limit port range.
# Default: 1024
#StreamMinPort 30000
# Default: 2048
#StreamMaxPort 32000

# Maximum number of threads running at the same time.
# Default: 10
MaxThreads 50

# Waiting for data from a client socket will timeout after this time (seconds).
```



## Vírus-scan: Solução para checagem de vírus

```
# Default: 120
ReadTimeout 300

# This option specifies the time (in seconds) after which clamd should
# timeout if a client doesn't provide any initial command after connecting.
# Default: 5
#CommandReadTimeout 5

# This option specifies how long to wait (in milliseconds) if the send buffer is full.
# Keep this value low to prevent clamd hanging
#
# Default: 500
#SendBufTimeout 200

# Maximum number of queued items (including those being processed by MaxThreads threads)
# It is recommended to have this value at least twice MaxThreads if possible.
# WARNING: you shouldn't increase this too much to avoid running out of file descriptors,
# the following condition should hold:
# MaxThreads*MaxRecursion + (MaxQueue - MaxThreads) + 6 < RLIMIT_NOFILE (usual max is
# 1024)
#
# Default: 100
#MaxQueue 200

# Waiting for a new job will timeout after this time (seconds).
# Default: 30
#IdleTimeout 60

# Don't scan files and directories matching regex
# This directive can be used multiple times
# Default: scan all
#ExcludePath ^/proc/
#ExcludePath ^/sys/

# Maximum depth directories are scanned at.
# Default: 15
#MaxDirectoryRecursion 20

# Follow directory symlinks.
# Default: no
#FollowDirectorySymlinks yes

# Follow regular file symlinks.
# Default: no
```



## Vírus-scan: Solução para checagem de vírus

```
#FollowFileSymlinks yes

# Scan files and directories on other filesystems.
# Default: yes
#CrossFilesystems yes

# Perform a database check.
# Default: 600 (10 min)
#SelfCheck 600

# Execute a command when virus is found. In the command string %v will
# be replaced with the virus name.
# Default: no
#VirusEvent /usr/local/bin/send_sms 123456789 "VIRUS ALERT: %v"

# Run as another user (clamd must be started by root for this option to work)
# Default: don't drop privileges
User clamav

# Initialize supplementary group access (clamd must be started by root).
# Default: no
AllowSupplementaryGroups yes

# Stop daemon when libclamav reports out of memory condition.
#ExitOnOOM yes

# Don't fork into background.
# Default: no
#Foreground yes

# Enable debug messages in libclamav.
# Default: no
#Debug yes

# Do not remove temporary files (for debug purposes).
# Default: no
#LeaveTemporaryFiles yes

# Detect Possibly Unwanted Applications.
# Default: no
#DetectPUA yes

# Exclude a specific PUA category. This directive can be used multiple times.
# See http://www.clamav.net/support/pua for the complete list of PUA
```



## Vírus-scan: Solução para checagem de vírus

```
# categories.
# Default: Load all categories (if DetectPUA is activated)
#ExcludePUA NetTool
#ExcludePUA PWTool

# Only include a specific PUA category. This directive can be used multiple
# times.
# Default: Load all categories (if DetectPUA is activated)
#IncludePUA Spy
#IncludePUA Scanner
#IncludePUA RAT

# In some cases (eg. complex malware, exploits in graphic files, and others),
# ClamAV uses special algorithms to provide accurate detection. This option
# controls the algorithmic detection.
# Default: yes
#AlgorithmicDetection yes

##
## Executable files
##

# PE stands for Portable Executable - it's an executable file format used
# in all 32 and 64-bit versions of Windows operating systems. This option allows
# ClamAV to perform a deeper analysis of executable files and it's also
# required for decompression of popular executable packers such as UPX, FSG,
# and Petite. If you turn off this option, the original files will still be
# scanned, but without additional processing.
# Default: yes
ScanPE yes

# Executable and Linking Format is a standard format for UN*X executables.
# This option allows you to control the scanning of ELF files.
# If you turn off this option, the original files will still be scanned, but
# without additional processing.
# Default: yes
ScanELF yes

# With this option clamav will try to detect broken executables (both PE and
# ELF) and mark them as Broken.Executable.
# Default: no
DetectBrokenExecutables yes
```





## Vírus-scan: Solução para checagem de vírus

##

## Documents

##

# This option enables scanning of OLE2 files, such as Microsoft Office documents and .msi files.

# If you turn off this option, the original files will still be scanned, but without additional processing.

# Default: yes

ScanOLE2 yes

# With this option enabled OLE2 files with VBA macros, which were not detected by signatures will be marked as "Heuristics.OLE2.ContainsMacros".

# Default: no

#OLE2BlockMacros no

# This option enables scanning within PDF files.

# If you turn off this option, the original files will still be scanned, but without decoding and additional processing.

# Default: yes

ScanPDF yes

##

## Mail files

##

# Enable internal e-mail scanner.

# If you turn off this option, the original files will still be scanned, but without parsing individual messages/attachments.

# Default: yes

ScanMail yes

# Scan RFC1341 messages split over many emails.

# You will need to periodically clean up \$TemporaryDirectory/clamav-partial directory.

# WARNING: This option may open your system to a DoS attack.

# Never use it on loaded servers.

# Default: no

#ScanPartialMessages yes

# With this option enabled ClamAV will try to detect phishing attempts by using



## Vírus-scan: Solução para checagem de vírus

```
# signatures.  
# Default: yes  
#PhishingSignatures yes  
  
# Scan URLs found in mails for phishing attempts using heuristics.  
# Default: yes  
#PhishingScanURLs yes  
  
# Always block SSL mismatches in URLs, even if the URL isn't in the database.  
# This can lead to false positives.  
#  
# Default: no  
#PhishingAlwaysBlockSSLMismatch no  
  
# Always block cloaked URLs, even if URL isn't in database.  
# This can lead to false positives.  
#  
# Default: no  
#PhishingAlwaysBlockCloak no  
  
# Allow heuristic match to take precedence.  
# When enabled, if a heuristic scan (such as phishingScan) detects  
# a possible virus/phish it will stop scan immediately. Recommended, saves CPU  
# scan-time.  
# When disabled, virus/phish detected by heuristic scans will be reported only at  
# the end of a scan. If an archive contains both a heuristically detected  
# virus/phish, and a real malware, the real malware will be reported  
#  
# Keep this disabled if you intend to handle "*.Heuristics.*" viruses  
# differently from "real" malware.  
# If a non-heuristically-detected virus (signature-based) is found first,  
# the scan is interrupted immediately, regardless of this config option.  
#  
# Default: no  
#HeuristicScanPrecedence yes  
  
##  
## Data Loss Prevention (DLP)  
##  
  
# Enable the DLP module  
# Default: No  
#StructuredDataDetection yes
```



## Vírus-scan: Solução para checagem de vírus

```
# This option sets the lowest number of Credit Card numbers found in a file
# to generate a detect.
# Default: 3
#StructuredMinCreditCardCount 5
```

```
# This option sets the lowest number of Social Security Numbers found
# in a file to generate a detect.
# Default: 3
#StructuredMinSSNCount 5
```

```
# With this option enabled the DLP module will search for valid
# SSNs formatted as xxx-yy-zzzz
# Default: yes
#StructuredSSNFormatNormal yes
```

```
# With this option enabled the DLP module will search for valid
# SSNs formatted as xxxyyzzzz
# Default: no
#StructuredSSNFormatStripped yes
```

```
##
## HTML
##
```

```
# Perform HTML normalisation and decryption of MS Script Encoder code.
# Default: yes
# If you turn off this option, the original files will still be scanned, but
# without additional processing.
#ScanHTML yes
```

```
##
## Archives
##
```

```
# ClamAV can scan within archives and compressed files.
# If you turn off this option, the original files will still be scanned, but
# without unpacking and additional processing.
# Default: yes
ScanArchive yes
```

```
# Mark encrypted archives as viruses (Encrypted.Zip, Encrypted.RAR).
# Default: no
```



## Vírus-scan: Solução para checagem de vírus

ArchiveBlockEncrypted no

##

## Limits

##

# The options below protect your system against Denial of Service attacks  
# using archive bombs.

# This option sets the maximum amount of data to be scanned for each input file.  
# Archives and other containers are recursively extracted and scanned up to this  
# value.

# Value of 0 disables the limit

# Note: disabling this limit or setting it too high may result in severe damage  
# to the system.

# Default: 100M

#MaxScanSize 150M

# Files larger than this limit won't be scanned. Affects the input file itself  
# as well as files contained inside it (when the input file is an archive, a  
# document or some other kind of container).

# Value of 0 disables the limit.

# Note: disabling this limit or setting it too high may result in severe damage  
# to the system.

# Default: 25M

#MaxFileSize 30M

# Nested archives are scanned recursively, e.g. if a Zip archive contains a RAR  
# file, all files within it will also be scanned. This options specifies how  
# deeply the process should be continued.

# Note: setting this limit too high may result in severe damage to the system.

# Default: 16

#MaxRecursion 10

# Number of files to be scanned within an archive, a document, or any other  
# container file.

# Value of 0 disables the limit.

# Note: disabling this limit or setting it too high may result in severe damage  
# to the system.

# Default: 10000

#MaxFiles 15000



## Vírus-scan: Solução para checagem de vírus

```
##
## Clamuko settings
##

# Enable Clamuko. Dazuko must be configured and running. Clamuko supports
# both Dazuko (/dev/dazuko) and DazukoFS (/dev/dazukofs.ctrl). DazukoFS
# is the preferred option. For more information please visit www.dazuko.org
# Default: no
#ClamukoScanOnAccess yes

# The number of scanner threads that will be started (DazukoFS only).
# Having multiple scanner threads allows Clamuko to serve multiple
# processes simultaneously. This is particularly beneficial on SMP machines.
# Default: 3
#ClamukoScannerCount 3

# Don't scan files larger than ClamukoMaxFileSize
# Value of 0 disables the limit.
# Default: 5M
#ClamukoMaxFileSize 10M

# Set access mask for Clamuko (Dazuko only).
# Default: no
#ClamukoScanOnOpen yes
#ClamukoScanOnClose yes
#ClamukoScanOnExec yes

# Set the include paths (all files inside them will be scanned). You can have
# multiple ClamukoIncludePath directives but each directory must be added
# in a separate line. (Dazuko only)
# Default: disabled
#ClamukoIncludePath /home
#ClamukoIncludePath /students

# Set the exclude paths. All subdirectories are also excluded. (Dazuko only)
# Default: disabled
#ClamukoExcludePath /home/bofh

# With this option you can whitelist specific UIDs. Processes with these UIDs
# will be able to access all files.
# This option can be used multiple times (one per line).
# Default: disabled
#ClamukoExcludeUID 0
```



## Vírus-scan: Solução para checagem de vírus

```
# With this option enabled ClamAV will load bytecode from the database.
# It is highly recommended you keep this option on, otherwise you'll miss detections for many new
# viruses.
# Default: yes
#Bytecode yes

# Set bytecode security level.
# Possible values:
#   None - no security at all, meant for debugging. DO NOT USE THIS ON PRODUCTION
#   SYSTEMS
#   This value is only available if clamav was built with --enable-debug!
#   TrustSigned - trust bytecode loaded from signed .c[lv]d files,
#   insert runtime safety checks for bytecode loaded from other sources
#   Paranoid - don't trust any bytecode, insert runtime checks for all
# Recommended: TrustSigned, because bytecode in .cvd files already has these checks
# Note that by default only signed bytecode is loaded, currently you can only
# load unsigned bytecode in --enable-debug mode.
#
# Default: TrustSigned
#BytecodeSecurity TrustSigned

# Set bytecode timeout in milliseconds.
#
# Default: 5000
# BytecodeTimeout 1000
```



## Vírus-scan: Solução para checagem de vírus

### **ANEXO 4 – Teste da instalação da servidora de aplicação + componente**

Segue relação de formas de acionar o componente manualmente (linha de comando), útil para testes da instalação de anti-vírus na servidora e do componente *virus-scan*, sem a necessidade de implantar toda a aplicação na servidora.

É possível simular teste com detecção de assinatura de vírus, vide:  
[http://en.wikipedia.org/wiki/EICAR\\_test\\_file](http://en.wikipedia.org/wiki/EICAR_test_file)

**(a) java -cp mail-1.4.5.jar -jar virus-scan-1.0.1.jar path-arquivo**

**(b) java -cp mail-1.4.5.jar -jar virus-scan-1.0.1.jar path-arquivo numero-anti-virus-provider**

- java -cp /home/clayton/mail-1.4.5.jar -jar target/virus-scan-1.0.1.jar  
/developer/java/apache/apache-jmeter-2.9/bin/diploma.pdf

17/06/2013 15:42:18 br.unicamp.ccuec.ctmq.virusScan.impl.AbstractVirusScan loadConfig  
**GRAVE:**

**virusScan-conf.properties NÃO ENCONTRADO, USANDO CONFIG. DEFAULT!  
RECOMENDA-SE CONFIG. PARA ENVIO DE E-MAIL DE ALERTA!**

java.io.FileNotFoundException  
at

br.unicamp.ccuec.ctmq.virusScan.impl.AbstractVirusScan.loadConfig(AbstractVirusScan.java:59)  
at

br.unicamp.ccuec.ctmq.virusScan.impl.AbstractVirusScan.<clinit>(AbstractVirusScan.java:49)  
17/06/2013 15:42:18 br.unicamp.ccuec.ctmq.virusScan.impl.AbstractVirusScan testVirusFromFile  
INFO:

...scanning...

17/06/2013 15:42:18 br.unicamp.ccuec.ctmq.virusScan.impl.AbstractVirusScan testVirusFromFile  
**AVISO:**

**arquivo /developer/java/apache/apache-jmeter-2.9/bin/diploma.pdf CHECADO E LIMPO!!!**



## Vírus-scan: Solução para checagem de vírus

- `java -cp /home/clayton/mail-1.4.5.jar -jar target/virus-scan-1.0.1.jar  
src/test/resources/eicar.txt`

17/06/2013 15:42:29 br.unicamp.ccuec.ctmq.virusScan.impl.AbstractVirusScan loadConfig  
**GRAVE:**

**virusScan-conf.properties NÃO ENCONTRADO, USANDO CONFIG. DEFAULT!  
RECOMENDA-SE CONFIG. PARA ENVIO DE E-MAIL DE ALERTA!**

java.io.FileNotFoundException  
at  
br.unicamp.ccuec.ctmq.virusScan.impl.AbstractVirusScan.loadConfig(AbstractVirusScan.java:59)  
at  
br.unicamp.ccuec.ctmq.virusScan.impl.AbstractVirusScan.<clinit>(AbstractVirusScan.java:49)  
17/06/2013 15:42:29 br.unicamp.ccuec.ctmq.virusScan.impl.AbstractVirusScan testVirusFromFile  
INFO:

**...scanning...**

17/06/2013 15:42:29 br.unicamp.ccuec.ctmq.virusScan.impl.AbstractVirusScan printScanStatus

**AVISO:**

====[RESULTADO]=====

scan(InputStream)  
string...: stream: Eicar-Test-Signature FOUND  
signature: Eicar-Test-Signature  
status...: FAILED  
exception: null  
=====

17/06/2013 15:42:29 br.unicamp.ccuec.ctmq.virusScan.impl.AbstractVirusScan testVirusFromFile  
**AVISO:**

**arquivo src/test/resources/eicar.txt VIRUS DETECTADO!!!**