INF-527 – Administração da segurança de Redes de Computadores

Relatório do Projeto

Marcelo Russi Mergulhão
Mario A. Franco Junior
Tiago Felipe Gonçalves

Campinas

2015

Introdução

Este projeto consiste na criação de um ambiente virtual que simula uma rede de servidores e clientes, com o intuito de aprender a instalar e configurar as diversas ferramentas estudadas em sala de aula, similar ao que foi feito na disciplina INF-520. Desta vez, entretanto, o foco será dado em segurança e as ferramentas devem implementar aspectos como uso de criptografia e assinaturas.

O projeto foi desenvolvido com a ajuda do ambiente de colaboração github (www.github.com), sendo criado utilizando o usuário de um dos integrantes, podendo ser localizado em https://github.com/marph13. Neste repostitório estão presentes as configurações de cada um dos serviços implementados, e portanto este documento apresentará a menor quantidade possível de trechos dos arquivos de configuração, indicando sua localização.

Topologia

A topologia escolhida para o ambiente da rede “inf520” pode ser ilustrado pela seguinte figura:

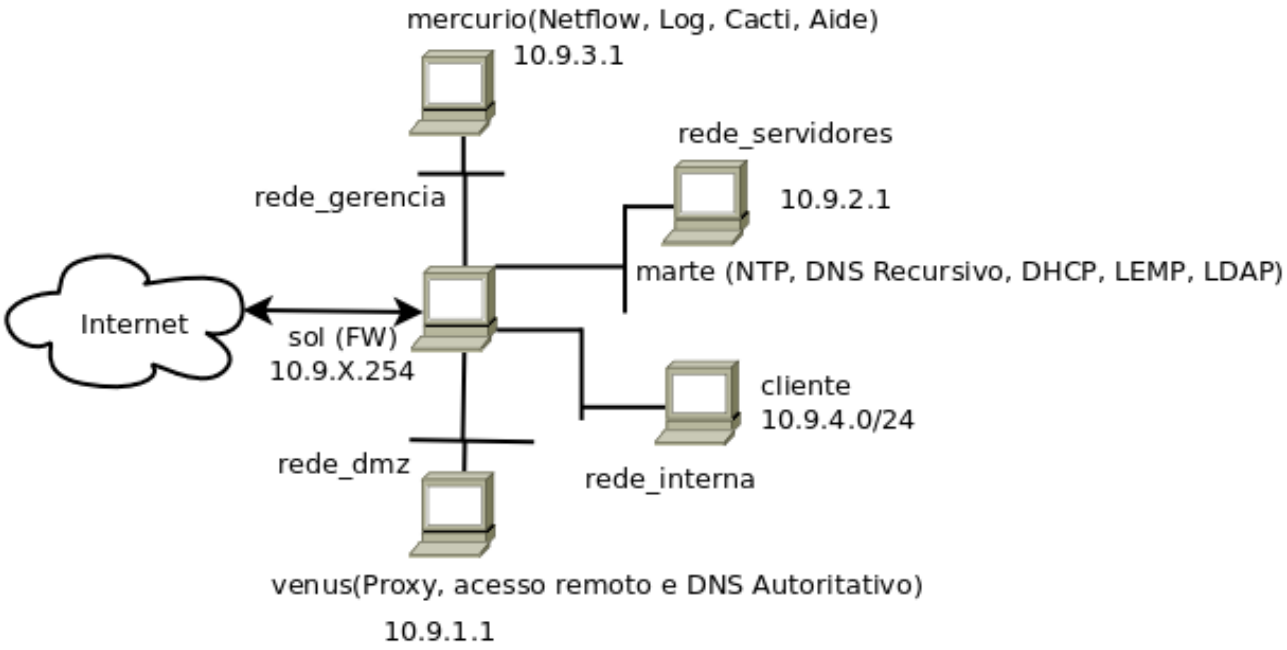

Figura 1 – Topologia de rede

Podemos observar que existem 4 redes conectadas ao nosso firewall (sol), provendo a divisão entre os serviços acessíveis remotamente da internet na “rede_dmz”, a rede de clientes como “rede_interna”, a rede de serviços como “rede_servidores” e a gerência do ambiente concentrado na “rede_gerencia”. As redes foram separadas para prover acesso apenas a quem deve conseguir acessar e quem fará este papel de filtro é nosso firewall.

Colocou-se apenas uma máquina em cada rede por economia de recursos, mas o ideal seria que houvesse uma maior separação dos serviços.

Além disso, por simplicidade de desenvolvimento, todos os servidores possuem uma interface dedicada de conexão com a máquina do desenvolvedor (host-only), que é a responsável por executar todas as máquinas virtuais e também por prover o acesso à internet ao gateway. A seguir descrevemos máquina a máquina os serviços presentes e indicamos a configuração aplicada.

Servidores

Sol (firewall)

Este é o firewall da nossa solução, responsável por filtrar os pacotes com destino indevido. Aqui temos a funcionalidade de filtro de pacotes e NAT, além de fazer um relay de DHCP da rede de clientes para o servidor DHCP na rede de servidores. Utilizamos como sistema operacional aqui o OpenBSD.

Todos os arquivos mencionados nesta seção se encontram no caminho “vm_config/sol” do repositório.

NAT e Firewall

As configurações de NAT e Firewall foram feitas utilizando regras de pf, contidas no arquivo pf.conf. Para acesso externo, foram liberados apenas os fluxos relacionados aos serviços presentes na DMZ (OpenVPN, proxy web, ssh e DNS Autoritativo), e apenas para esta rede específica. Como alguns serviços da DMZ necessitam acessar serviços internos, como acesso a DNS recursivo, autenticação e encaminhamento de requisições Web, foram liberados entre a DMZ e a rede de servidores tais serviços também. A rede de gerência possui acesso a todas as demais, mas o acesso a ela é bloqueado quando iniciado de outras redes. A rede interna pode acessar as redes DMZ e de servidores nos serviços disponibilizados, mas os acessos a ela também são barrados.

Além disso, colocamos nas regras de firewall pontos específicos para tratar os “bogons”, que são Ips reservados e que não poderiam chegar ao nosso servidor pela internet.

Para testar o firewall realizamos varreduras com nmap a partir da máquina rodando as Vms numa interface host-only para simular o acesso externo, primeiramente um scan genérico (que só mostra a porta 22 aberta, pois usamos o ssh no desenvolvimento) e posteriormente um scan específico para garantir a passagem da porta de openvpn.

```
root@marceloNoteLinux:/home/marcelo# nmap 192.168.56.254

Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-12 15:25 BRST
Nmap scan report for sol (192.168.56.254)
Host is up (0.00082s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 08:00:27:0D:C7:16 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 5.94 seconds
```

```
root@marceloNoteLinux:/home/marcelo# nmap 192.168.56.254 -p 1194 -sU

Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-12 15:22 BRST
Nmap scan report for sol (192.168.56.254)
Host is up (0.00051s latency).
PORT     STATE         SERVICE
1194/udp open|filtered openvpn
```

```
MAC Address: 08:00:27:0D:C7:16 (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 0.54 seconds
```

```
root@marte:/home/base# nmap -p 25 mail.google.com

Starting Nmap 6.40 ( http://nmap.org ) at 2015-12-12 15:27 BRST
Nmap scan report for mail.google.com (173.194.118.21)
Host is up (0.011s latency).
Other addresses for mail.google.com (not scanned): 173.194.118.22
rDNS record for 173.194.118.21: gru06s09-in-f21.1e100.net
PORT   STATE    SERVICE
25/tcp filtered smtp

Nmap done: 1 IP address (1 host up) scanned in 1.36 seconds
```

Um ponto digno de nota é que realizamos a gerência da porta 25: uma vez que não temos servidor de e-mail interno e escolhemos utilizar uma solução na nuvem (gmail), não precisamos liberar o acesso à porta 25. Isso pode ser verificado no teste de firewall, que mostra a porta como bloqueada tanto para hosts internos com destino à porta 25 de algum outro host quanto para o acesso de hosts externos a qualquer host interno. O acesso à plataforma na nuvem será feita por HTTPs no cliente web disponibilizado pelo próprio Google.

DHCP Relay

Para o papel de dhcprelay tivemos problemas com um bug do “dhcrelay” padrão para o DHCPv6, então utilizamos o software wide-dhcpv6, que pode ser obtido em http://openbsd.c3sl.ufpr.br/pub/OpenBSD/5.8/packages/i386/wide-dhcpv6-20080615p3.tgz. A configuração deste serviço é bem simples e consiste apenas em apontar quais as interfaces relacionadas ao redirecionamento de pedidos de endereço via DHCP.

```
/usr/local/sbin/dhcrelay -i em2 10.9.2.1
/usr/local/sbin/dhcrelay -i em4 10.9.2.1
/usr/local/sbin/dhcp6relay -r em4 -s fc00:10:9:2::1 em2
```

O teste realizado para este serviço será a obtenção com sucesso de um IP via DHCP no cliente.

Marte (Serviços internos)

DNS Recursivo

Utilizou-se o unbound como servidor DNS recursivo, encaminhando as consultas ao domínio “inf527.org” para o servidor venus, presente na DMZ, enquanto as demais foram encaminhadas para o DNS de endereço 8.8.8.8. Durante os testes na Unicamp o DNS utilizado foi o da própria Unicamp.

O arquivo relacionado com o servidor DNS é “unbound.conf”, que deve estar presente no

caminho “/etc/unbound/unbound.conf” para ser carregado pelo software.

O teste deste serviço deu-se com o comando dig, perguntando sobre o IPv4, IPv6 e reversos de cada um dos servidores da topologia. Os comandos foram dados a partir da VM cliente e as consultas foram direcionadas ao servidor recursivo, que deveria consultar o servidor autoritativo e então retornar. Os resultados obtidos foram:

IPv4

```
base@cliente:~$ dig +noall +answer sol.inf527.org
sol.inf527.org.         3600    IN      A       10.9.0.254
base@cliente:~$ dig +noall +answer venus.inf527.org
venus.inf527.org.       3600    IN      A       10.9.1.1
base@cliente:~$ dig +noall +answer marte.inf527.org
marte.inf527.org.       3600    IN      A       10.9.2.1
base@cliente:~$ dig +noall +answer mercurio.inf527.org
mercurio.inf527.org.    3600    IN      A       10.9.3.1
```

Reverso IPv4

```
base@cliente:~$ dig -x 10.9.1.1 +noall +answer
1.1.9.10.in-addr.arpa.  3459    IN      PTR     venus.9.10.in-addr.arpa.
base@cliente:~$ dig -x 10.9.2.1 +noall +answer
1.2.9.10.in-addr.arpa.  3459    IN      PTR     marte.9.10.in-addr.arpa.
base@cliente:~$ dig -x 10.9.3.1 +noall +answer
1.3.9.10.in-addr.arpa.  3600    IN      PTR     mercurio.9.10.in-addr.arpa.
base@cliente:~$ dig -x 10.9.0.254 +noall +answer
254.0.9.10.in-addr.arpa. 3600   IN      PTR     sol.9.10.in-addr.arpa.
```

IPv6

```
base@cliente:~$ dig AAAA marte.inf527.org +noall +answer
marte.inf527.org.       3600    IN      AAAA    fc00:10:9:2::1
base@cliente:~$ dig AAAA venus.inf527.org +noall +answer
venus.inf527.org.       3600    IN      AAAA    fc00:10:9:1::1
base@cliente:~$ dig AAAA mercurio.inf527.org +noall +answer
mercurio.inf527.org.    3600    IN      AAAA    fc00:10:9:3::1
base@cliente:~$ dig AAAA sol.inf527.org +noall +answer
sol.inf527.org.         3600    IN      AAAA    fc00:10:9::254
```

Reverso IPv6

```
base@cliente:~$ dig -x fc00:10:9:1::1 +noall +answer
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.9.0.0.0.0.1.0.0.0.0.c.f.ip6.arpa. 2785 IN PTR
venus.0.0.0.9.0.0.0.0.1.0.0.0.0.c.f.ip6.arpa.
base@cliente:~$ dig -x fc00:10:9:2::1 +noall +answer
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.0.0.9.0.0.0.0.1.0.0.0.0.c.f.ip6.arpa. 2781 IN PTR
marte.0.0.0.9.0.0.0.0.1.0.0.0.0.c.f.ip6.arpa.
base@cliente:~$ dig -x fc00:10:9:3::1 +noall +answer
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.0.0.9.0.0.0.0.1.0.0.0.0.c.f.ip6.arpa. 3600 IN PTR
mercurio.0.0.0.9.0.0.0.0.1.0.0.0.0.c.f.ip6.arpa.
base@cliente:~$ dig -x fc00:10:9:0::254 +noall +answer
4.5.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.9.0.0.0.0.1.0.0.0.0.c.f.ip6.arpa. 3600 IN PTR
sol.0.0.0.9.0.0.0.0.1.0.0.0.0.c.f.ip6.arpa.
```

DHCP

Utilizou-se o software ISC-DHCP para criar um servidor de DHCPv4 e outro de DHCPv6, ambos para fornecimento de endereços dinâmicos aos clientes. A faixa de endereços utilizada foi 10.9.4.10 a 10.9.4.20 para IPv4 e fc00:10:9:4:10 a fc00:10:9:4::20 para IPv6, enquanto o servidor DHCP utilizou os endereços 10.9.2.1 e fc00:10:9:2::1. Como as redes são diferentes, é necessário um relay DHCP, que foi implementado segundo o descrito na seção da VM sol.

Os arquivos de configuração são “dhcp.conf”e “dhcp6.conf”, ambos posicionados em “/etc/dhcp” e referenciados na subida do servidor. O teste de funcionamento se deu através da observação da máquina cliente, configurada para requisitar endereços IPv4 e IPv6 por DHCP. O resultado foi a obtenção dos seguintes endereços:

```
base@cliente:~$ ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 08:00:27:8a:e1:32
          inet addr:10.9.4.10  Bcast:10.9.4.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe8a:e132/64 Scope:Link
          inet6 addr: fc00:10:9:4::20/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:369 errors:0 dropped:0 overruns:0 frame:0
          TX packets:450 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:83967 (83.9 KB)  TX bytes:56833 (56.8 KB)
```

NFS

O objetivo da solução ter NFS é fazer com que os usuários possuam seus arquivos num diretório NFS e possam acessá-los em qualquer máquina da rede de clientes. O NFS foi implementado com a utilização do programa “nfsd”, cujos arquivos de configuração são: “exports”, indicando quais diretório exportar e com qual permissão, “hosts.allow” para permitir acesso a determinados hosts e “hosts.deny” para negar acesso. Permitiu-se o acesso ao sistema de arquivos apenas para os servidores da rede de clientes e o diretório exportado foi o “/home/users”, criado específicamente para esse objetivo e que terá os diretórios home dos usuários criados conforme eles se autenticarem no LDAP.

O teste do serviço de NFS foi feito através da montagem do diretório na máquina cliente com a inserção da instrução em “/etc/fstab”, criação de um arquivo e observação do mesmo na VM

marte:

VM cliente:

```
$ pwd
/home/users/cnumero1
$ mount
/dev/sda1 on / type ext4 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
none on /sys/fs/cgroup type tmpfs (rw)
none on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
udev on /dev type devtmpfs (rw,mode=0755)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
none on /run/lock type tmpfs (rw,noexec,nosuid,nodev,size=5242880)
none on /run/shm type tmpfs (rw,nosuid,nodev)
none on /run/user type tmpfs (rw,noexec,nosuid,nodev,size=104857600,mode=0755)
none on /sys/fs/pstore type pstore (rw)
rpc_pipefs on /run/rpc_pipefs type rpc_pipefs (rw)
systemd on /sys/fs/cgroup/systemd type cgroup (rw,noexec,nosuid,nodev,none,name=systemd)
10.9.2.1:/home/users/ on /home/users type nfs (rw,vers=4,addr=10.9.2.1,clientaddr=10.9.4.10)
```

VM Marte:

```
base@marte:/home$ ls -l users/
total 4
drwxr-xr-x 3 5000 clientes 4096 Dec  8 14:33 cnumero1
```

LEMP (Nginx + MySQL + PHP)

Nesta máquina virtual temos um ambiente de servidor Web com banco de dados e tratamento dinâmico de páginas integrado.

Escolhemos o banco de dados MySQL como solução de armazenamento dos dados, o servidor Web nginx e a linguagem PHP para tratamento de conteúdo dinâmico.

O acesso ao servidor web será feito via https, portanto foi necessária a criação de um certificado autoassinado para ele. Além disso, tivemos que alterar as configurações para apontar corretamente para os arquivos, limitar o acesso apenas aos endereços conhecidos (como index.html) e usar a porta 443.

O teste de funcionamento da solução LEMP será o acesso de uma máquina cliente ao servidor, pedindo pela página de informações info.php. Este endereço deve fornecer uma visão das configurações do servidor, e não deveria estar disponível num ambiente em produção, podendo ser usado apenas para verificar funcionalidades.

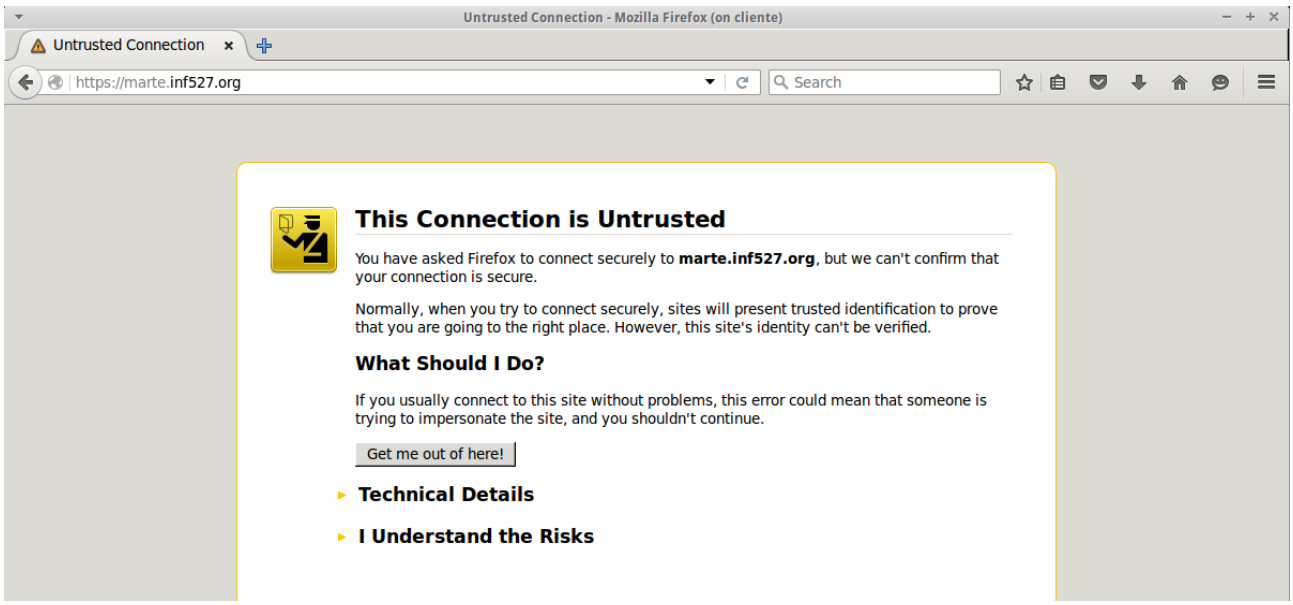

Figura 2 – Verificação de certificado nginx

Figura 3 – Acesso ao índice do servidor Web

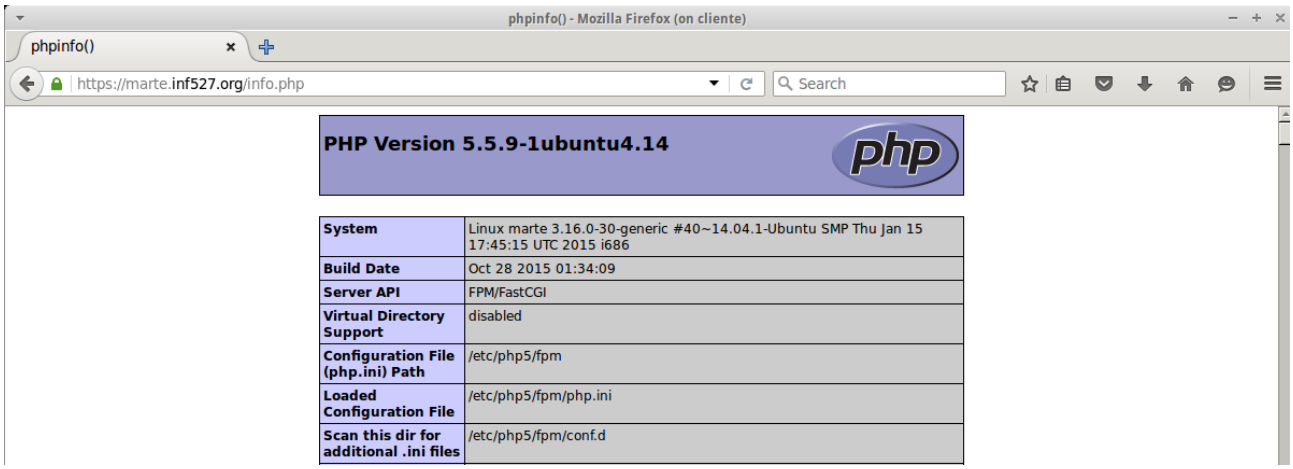

Figura 4 – Acesso ao link de informações PHP

LDAP

Como sistema de acesso centralizado utilizamos o OpenLDAP e como ferramenta de administração do banco de usuários utilizamos phpldapadmin.

A estrutura utilizada foi bastante simples, apenas para demonstrar como realizar a separação

de usuários e o acesso utilizando autenticação centralizada. A figura a seguir demonstra na ferramenta phpldapadmin o banco utilizado:

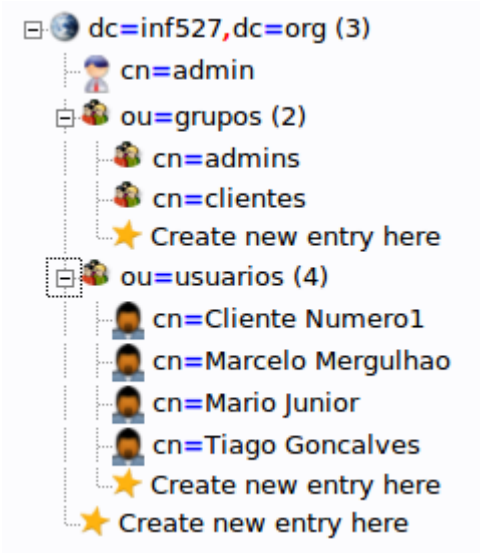

Figura 5 – Usuários e grupos na ferramenta phpldapadmin

Podemos observar que a organização foi feita colocando os usuários em uma OU específica (usuarios) e os grupos aos quais estes usuários pertencem em outra OU. Os usuários relacionados aos autores deste trabalho estão como “admins” e o cliente é representado por “Cliente Numero1”. O cliente deve conseguir acessar a máquina cliente com suas credenciais centralizadas e então terá o seu home via NFS.

O servidor LDAP foi configurado para prover autenticação com TLS, com certificados autoassinados. Observando o tráfego de uma tentativa de acesso do cliente 10.9.4.10 podemos observar o início da sessão TLS e posterior tráfego de dados criptografados, demonstrado na Figura a seguir:

```
30 0.327052000 10.9.4.10    10.9.2.1     TLSv1.2    183 Client Hello
31 0.327082000 10.9.2.1     10.9.4.10    TCP         66 ldaps > 38185 [ACK] Seq=1 Ack=118 Win=28960 Len=0 TSval=2034887 TSecr=1024134
32 0.327777000 10.9.2.1     10.9.4.10    TLSv1.2    152 Server Hello
33 0.327943000 10.9.2.1     10.9.4.10    TLSv1.2   1664 Certificate
34 0.328118000 10.9.2.1     10.9.4.10    TLSv1.2     75 Server Hello Done
35 0.329847000 10.9.4.10    10.9.2.1     TCP         66 38185 > ldaps [ACK] Seq=118 Ack=87 Win=29216 Len=0 TSval=1024135 TSecr=2034887
36 0.329882000 10.9.4.10    10.9.2.1     TCP         66 38185 > ldaps [ACK] Seq=118 Ack=1685 Win=32416 Len=0 TSval=1024135 TSecr=2034887
37 0.329892000 10.9.4.10    10.9.2.1     TCP         66 38185 > ldaps [ACK] Seq=118 Ack=1694 Win=32416 Len=0 TSval=1024135 TSecr=2034887
38 0.331934000 10.9.4.10    10.9.2.1     TLSv1.2    205 Client Key Exchange
39 0.331952000 10.9.4.10    10.9.2.1     TLSv1.2     72 Change Cipher Spec
40 0.333234000 10.9.4.10    10.9.2.1     TLSv1.2    311 Encrypted Handshake Message
41 0.338014000 10.9.2.1     10.9.4.10    TCP         66 ldaps > 38185 [ACK] Seq=1694 Ack=508 Win=31104 Len=0 TSval=2034890 TSecr=1024135
42 0.338293000 10.9.2.1     10.9.4.10    TLSv1.2     72 Change Cipher Spec
43 0.338571000 10.9.2.1     10.9.4.10    TLSv1.2    247 Encrypted Handshake Message
44 0.340310000 10.9.4.10    10.9.2.1     TCP         66 38185 > ldaps [ACK] Seq=508 Ack=1881 Win=35296 Len=0 TSval=1024137 TSecr=2034890
45 0.340487000 10.9.4.10    10.9.2.1     TLSv1.2    199 Application Data
```

Figura 6 – Tráfego TLS entre LDAP e cliente

Venus

Este servidor tem o intuito de servir como ponto de acesso dos serviços disponibilizados pela internet, contendo um proxy para o servidor Web, o servidor OpenVPN e SSH, além do DNS autoritativo.

DNS Autoritativo

Utilizamos o NSD como servidor autoritativo, configurando o nosso domínio com o nome “inf527.org”. Diferentemente do servidor recursivo, que apenas aponta para um servidor autoritativo, este possui os registros de cada um dos nomes que pode resolver, que no nosso caso ficaram concentrados em 3 arquivos: “/etc/nsd/inf527.org.zone”, que contém os registros A (IPv4) e AAAA(IPv6) para todas as máquinas e um CNAME para nosso servidor web proxy(www); O segundo arquivo é “/etc/nsd/inf527.org.reverso.zone” e diz respeito à resolução de endereços IPv4 reversos; Similarmente, “/etc/nsd/inf527.org.reverso_v6.zone” cuida da resolução de IPv6 reversos.

O NSD utiliza as zonas citadas através de entradas do tipo “zone”, presentes em seu arquivo de configuração, “nsd.conf”. O teste do servidor DNS autoritativo é o mesmo do servidor DNS recursivo, uma vez que o segundo só repassa as requisições para o primeiro.

Proxy

Para fazer o serviço de proxy e permitir que usuários na internet acessem nosso conteúdo Web, utilizamos o nginx. Para que o nginx realize a função de proxy, precisamos apenas configurar o site habilitado (o “default” modificado, para nosso caso) adicionando uma referência de proxy no grupo “location”.

```
location / {
    proxy_set_header        Host $host;
     proxy_set_header        X-Real-IP $remote_addr;
     proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
     proxy_set_header        X-Forwarded-Proto $scheme;
    proxy_pass          https://marte.inf527.org;
   }
```

Dessa forma, o nosso servidor sabe que deve encaminhar os pedidos que chegarem para o servidor marte.inf527.org, que contém nosso servidor Web de fato.

O acesso ao servidor proxy também será feito com http, portanto ele também possui certificados próprios e as propriedades relacionadas a esse tipo de acesso, como “ssl on”.

Podemos verificar que o servidor proxy está funcionando quando tentamos acessar o “www” a partir da máquina cliente, por exemplo. Esse não é o caminho padrão de acesso, mas serve para validar que nosso proxy está de fato funcionando.

Figura 7 – Acesso ao Web Server via Proxy

OpenVPN

O OpenVPN é software voltado à criação de redes virtuais entre um servidor e um cliente, permitindo que usuários externos à organização realizem acessos como se estivessem conectados internamente. O serviço utiliza a porta 1194 e TLS para prover a confidencialidade da comunicação. As configurações do OpenVPN são feitas em par, no servidor com "/etc/openvpn/openvpn.conf" e "/etc/openvpn/auth-ldap.conf" e no cliente com uma especificação da VPN feita em "unicamp.ovpn", além dos certificados de ambas as máquinas.

O teste de funcionamento do OpenVPN foi feito com conexão orientada a interface host only (aplicamos as mesmas regras de firewall da interface externa na mesma somente para fins de simular uma wan), da maquina física fizemos a conexão no servidor venus (192.168.56.2) exigindo autenticação via ldaps (tls), sendo fornecido a conexão a rota 10.9.1.0/24 para uso.

```
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 TLS: Initial packet from [AF_INET]192.168.56.13:1194, sid=b5787e9c a9cef51d
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 PLUGIN_CALL: POST /usr/lib/openvpn/openvpn-auth-ldap.so/PLUGIN_AUTH_USER_PASS_VERIFY status=0
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 TLS: Username/Password authentication succeeded for username 'tgoncalves'
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 Data Channel Encrypt: Cipher 'BF-CBC' initialized with 128 bit key
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 Data Channel Encrypt: Using 160 bit message hash 'SHA1' for HMAC authentication
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 Data Channel Decrypt: Cipher 'BF-CBC' initialized with 128 bit key
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 Data Channel Decrypt: Using 160 bit message hash 'SHA1' for HMAC authentication
```

```
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-SHA
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 [] Peer Connection Initiated with [AF_INET]192.168.56.13:1194
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 MULTI_sva: pool returned IPv4=10.24.24.18, IPv6=(Not enabled)
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 PLUGIN_CALL: POST /usr/lib/openvpn/openvpn-auth-ldap.so/PLUGIN_CLIENT_CONNECT status=0
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 OPTIONS IMPORT: reading client specific options from: /tmp/openvpn_cc_7f1e96d866d420c416f3e71e004108c7.tmp
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 MULTI: Learn: 10.24.24.18 -> 192.168.56.13:1194
Dec 12 16:52:25 venus ovpn-openvpn[1182]: 192.168.56.13:1194 MULTI: primary virtual IP for 192.168.56.13:1194: 10.24.24.18
Dec 12 16:52:27 venus ovpn-openvpn[1182]: 192.168.56.13:1194 PUSH: Received control message: 'PUSH_REQUEST'
Dec 12 16:52:27 venus ovpn-openvpn[1182]: 192.168.56.13:1194 send_push_reply(): safe_cap=940
Dec 12 16:52:27 venus ovpn-openvpn[1182]: 192.168.56.13:1194 SENT CONTROL [UNDEF]: 'PUSH_REPLY,route 10.9.1.0 255.255.255.0,route 10.24.24.0 255.255.255.0,topology net30,ping 10,ping-restart 120,ifconfig 10.24.24.18 10.24.24.17' (status=1)
```

Mercurio

Gerenciamento Vms(Cacti, logstash e Kibana)

Utilizamos diversas ferramentas no gerenciamento de máquinas: Cacti para realizar as coletas de dados das Vms com snmp; Logstash para fazer o recebimento de logs; Kibana, para realizar a visualização de dados numa interface web; Elasticsearch para fazer o relacionamento dos dados.

O cacti controla a coleta de dados SNMP, mas utiliza um agente instalado em cada servidor, o “snmpd”. A configuração deste agente é feita com o arquivo “/etc/snmp/snmpd.conf” e que controla a porta do agente, o protocolo usado (TCP, UDP), a community e diversas outras características.

O teste de funcionamento do cacti foi a execução de seu ambiente web, que possui muitos gráficos pré-configurados do monitoramento de tráfego e recursos. Foram adicionadas monitorações em todos os servidores, sendo mostrada na figura a seguir apenas o host local, por simplicidade.

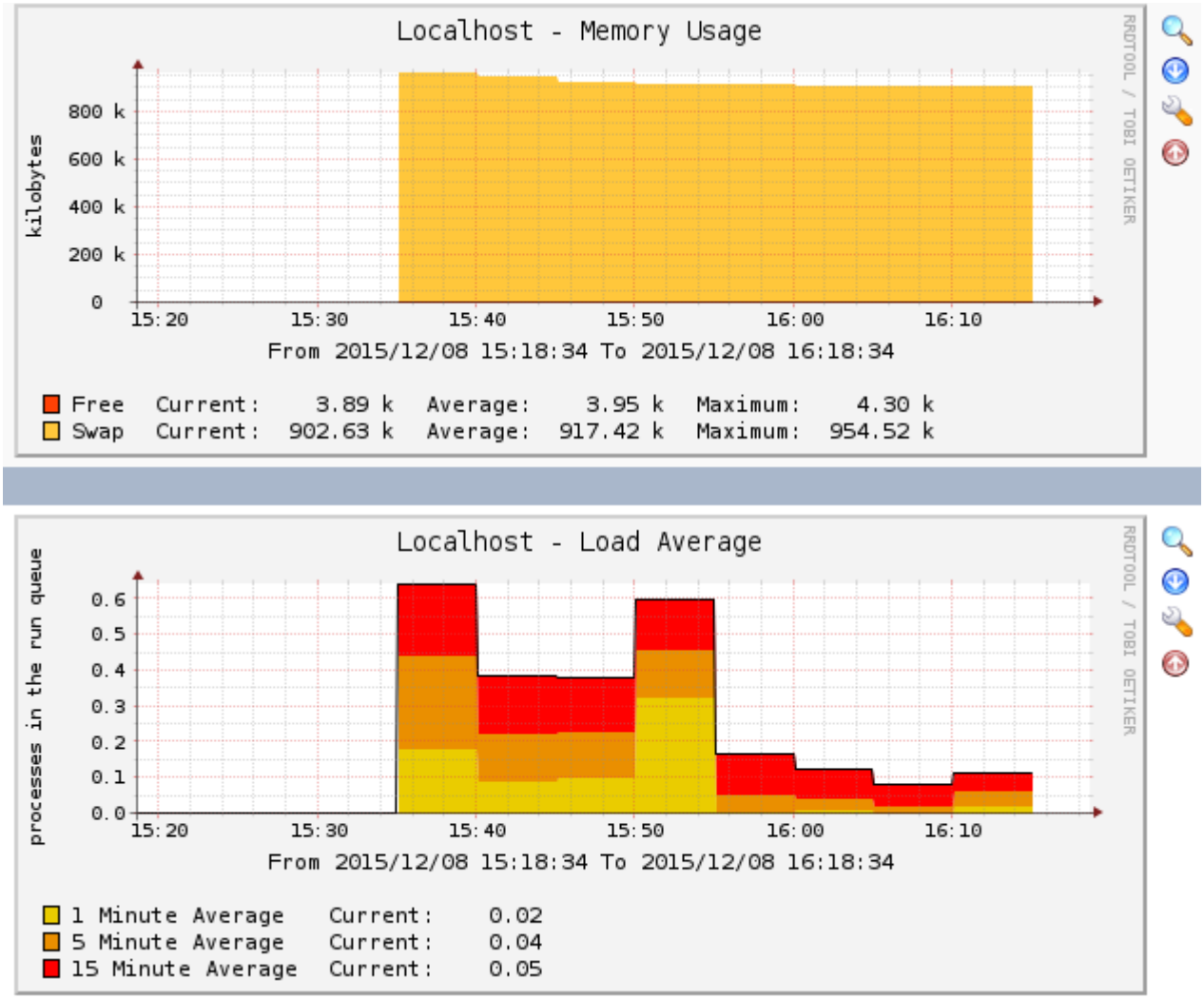

Figura 8 – Cacti e o monitoramento de recursos

O Logstash, Elasticsearch e o Kibana atuam em conjunto, sendo que o primeiro fica escutando a chegada de logs na porta 514 (tal qual o syslog), o Elasticsearch é uma feramenta de busca nos arquivos de log e o Kibana pode ser acessado pela porta 5601 para visualização dos logs. O kibana permite a criação de vários gráficos a partir de templates para a visualização dos dados. Um exemplo de dashboard criado com a ferramenta é o seguinte:

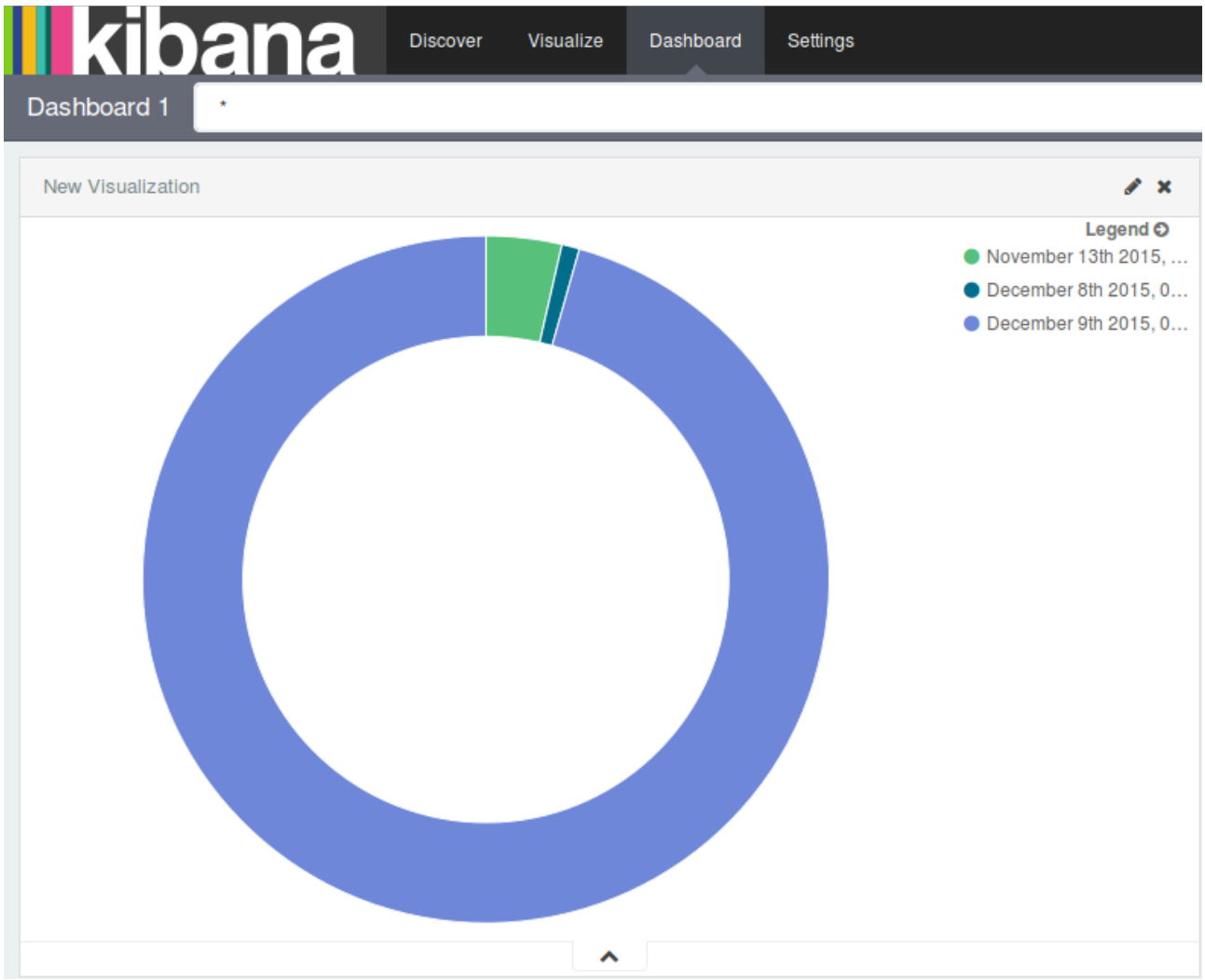

Figura 9 – Dashboard de Visualização dos logs com Kibana

Netflow (coleta de fluxo)

O serviço de netflow foi implementado com a aplicação nfdump, que utiliza o arquivo de configuração "/etc/default/nfdump". Esse servidor funciona apenas como receptor do tráfego coletado, cujo redirecionamento é feita no firewall (sol) com a ajuda do firewall (PF). A nossa configuração apenas indica para coletar o tráfego de todas as interfaces e enviar os dados para a mercurio.

Para a visualização dos dados de NetFlow foi instalado um programa chamado "nfsen", que possui uma interface web de visualização, permitindo uma visão e filtragem dos dados mais simplificada. Uma imagem da ferramenta em funcionamento está a seguir:

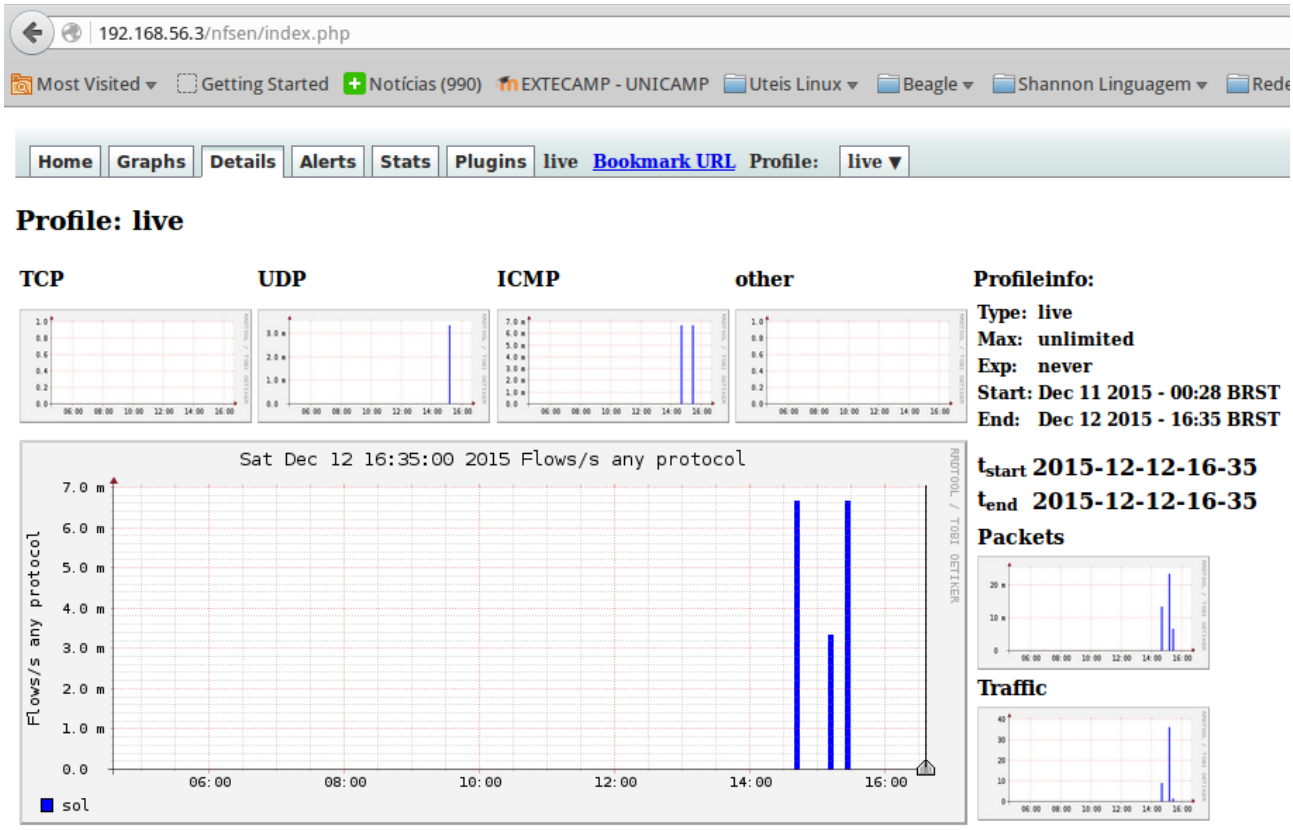

Figura 10 – Fluxos coletados com NetFlow no FlowView

O teste deste serviço é a própria visualização do fluxo, que pode ser verificado na imagem anterior.

Integridade de Arquivos

Apenas para fins demonstrativos, foi incluída na máquina mercurio uma ferramenta de verificação de integridade de arquivos chamada AIDE (Advanced Intrusion Detection Environment). O trabalho do AIDE é criar um banco de dados a partir dos arquivos de configuração contendo os valores em hash destes arquivos. Assim, quando um deles for alterado o valor comparado com a base original não será compatível e saberemos que houve um acesso, possivelmente indevido.

Para criar o banco executamos o comando “/usr/sbin/aideinit” e então será criado o arquivo “/var/lib/aide/aide.db.new”. Copiando o banco para “/var/lib/aide/aide.db” podemos manter a nossa referência, e então checagens posteriores podem ser feitas com “aide -c /etc/aide/aide.conf –check”.

Realizando a primeira checagem obtivemos:

```
base@mercurio:/home$ sudo aide -c /etc/aide/aide.conf --check
WARNING: Old db contains a entry that shouldn't be there, run --init or --update
AIDE 0.16a2-19-g16ed855 found NO differences between database and filesystem. Looks okay!!
Start timestamp: 2015-12-08 19:40:37 -0200
Verbose level: 6

Number of entries:      0
```

```
---------------------------------------------------
The attributes of the (uncompressed) database(s):
---------------------------------------------------

/var/lib/aide/aide.db
  RMD160   : csKuz/ycOqsXK4RaNk5VMEd4NO8=
  TIGER    : XhhIvNV/hld6I8AQ3jTVd8bCU9sVvWaY
  SHA256   : fN8t7zNEqtURVVgV3CgsVyAW95vDvt5Y
             F9rNjZs8lsg=
  SHA512   : 2SjpyVzmI6AY7rtE8RSbmLipvkRPbSXf
             g4y+UmK3S9dnw3Eslzm/kYJM1qkpck22
             9r9C9EO+Yvlscp+sJYShJw==
  CRC32    : 47nFAw==
  HAVAL    : kkOoGO8p3pmLlghuWg4Qi6dDPdMkwKYZ
             t0Fp1CCPd8Q=
  GOST     : PVwQU83ZC94iklsgslATNlT+loLujCpX
             NtpTSPaMVBE=


End timestamp: 2015-12-08 19:40:37 -0200 (run time: 0m 0s)
```