

Sessão 8: Auditoria de segurança da informação

1) Instalação do Nessus



Esta atividade será realizada na máquina virtual *KaliLinux-G*.

Nesta atividade iremos instalar e configurar o Tenable Nessus (<https://www.tenable.com/products/nessus-home>), um *scanner* de vulnerabilidades desenvolvido pela Tenable Network Security. O projeto era *open source* até a versão 2.2.11, em 2005, quando foi lançada a versão 3 do Nessus *engine* e ele se tornou, então, proprietário. O software ainda é gratuito para um bom número de usos, excluindo-se testes de *compliance* (como PCI, CIS e FDCC), auditorias de rede e checagens mais recentes, bem como algumas outras características.

O OpenVAS (<http://www.openvas.org/>) é um *fork open source* bastante popular do Nessus, que vem inclusive pré-instalado na distribuição Kali Linux.

1. Com a máquina *KaliLinux-G* **desligada**, acesse o menu *Settings > Storage*, clique na linha da controladora SATA e depois no pequeno símbolo de um HD com um + verde. Iremos adicionar um novo disco de 30 GB para armazenar a instalação do Nessus, já que o disco atual, de 20 GB, não será suficiente. Após clicar no ícone:
 - Selecione *Create new disk*.
 - Tipo do arquivo: mantenha *VDI*.
 - Tipo de alocação: mantenha *Dynamically allocated*.
 - Nome do disco: **kali-nessus**
 - Tamanho do disco: 30 GB

Além disso, será necessário voltar a máquina *KaliLinux-G* para a DMZ. Acesse *Settings > Network* e mude o nome do adaptador *host-only* da VM para o mesmo das máquinas *LinServer-G* e *WinServer-G*.

Ao final do processo, ligue a máquina *KaliLinux-G*.

2. Após o *boot*, faça login como usuário **root** e abra um terminal. Vamos particionar, formatar e montar o disco adicionado. Primeiro, descubra a letra sob a qual o disco foi detectado:

```
# dmesg | grep -i 'Attached SCSI disk'
[ 1.828729] sd 1:0:0:0: [sdb] Attached SCSI disk
[ 1.856784] sd 0:0:0:0: [sda] Attached SCSI disk
```

```
# fdisk -l /dev/sdb
Disk /dev/sdb: 30 GiB, 32212254720 bytes, 62914560 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Como era de se esperar, o disco foi detectado como `/dev/sdb`. Particione-o usando o `fdisk`: crie uma única partição primária, ocupando a totalidade do disco, com tipo de sistema de arquivos `Linux`.

```
# fdisk /dev/sdb

Welcome to fdisk (util-linux 2.31.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x986bc0aa.
```

```
Command (m for help): o
Created a new DOS disklabel with disk identifier 0xc0163032.
```

```
Command (m for help): n
Partition type
  p   primary (0 primary, 0 extended, 4 free)
  e   extended (container for logical partitions)
Select (default p):

Using default response p.
Partition number (1-4, default 1):
First sector (2048-62914559, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-62914559, default 62914559):

Created a new partition 1 of type 'Linux' and of size 30 GiB.
```

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Agora, formate o disco com o sistema de arquivos `ext4`:

```
# mkfs.ext4 /dev/sdb1
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 7864064 4k blocks and 1966080 inodes
Filesystem UUID: 99654695-1f56-4521-8cd5-da0c533b11ae
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

Iremos montar essa partição no **/opt**. Primeiro, monte-a temporariamente no diretório **/mnt** e faça o *backup* dos dados preexistentes no **/opt** para dentro dela, depois desfaça o *mount* temporário.

```
# mount /dev/sdb1 /mnt/
```

```
# rsync -av /opt/ /mnt/
```

```
# umount /mnt/
```

Descubra qual o UUID (*Universally Unique Identifier*) dessa nova partição. Em seguida, usando esse dado, crie uma nova linha no **/etc/fstab** que monte a partição automaticamente no diretório **/opt** durante o *boot*. Finalmente, monte-a usando **mount -a** e verifique o funcionamento da sua configuração.

```
# blkid | grep '^/dev/sdb1' | cut -d' ' -f2 | sed 's/"//g'
UUID=99654695-1f56-4521-8cd5-da0c533b11ae
```

```
# uuid=$( blkid | grep '^/dev/sdb1' | cut -d' ' -f2 | sed 's/"//g' ); echo "$uuid
/opt ext4 defaults 0 2" >> /etc/fstab; unset uuid
```

```
# tail -n1 /etc/fstab
UUID=99654695-1f56-4521-8cd5-da0c533b11ae /opt ext4 defaults 0 2
```

```
# mount -a
```

```
# mount | grep '^/dev/sdb1 '  
/dev/sdb1 on /opt type ext4 (rw,relatime)
```

3. Vamos reconfigurar a rede da máquina *KaliLinux-G* para a DMZ. Edite o arquivo `/etc/network/interfaces` como se segue:

```
# nano /etc/network/interfaces  
(...)
```

```
# cat /etc/network/interfaces  
source /etc/network/interfaces.d/*  
  
auto lo  
iface lo inet loopback  
  
auto eth0  
iface eth0 inet static  
address 172.16.1.30/24  
gateway 172.16.1.1
```

```
# systemctl restart networking
```

4. O próximo passo é fazer o download do pacote do Nessus. Na máquina *KaliLinux-G*, acesse a URL <https://www.tenable.com/products/nessus-home> com o navegador Firefox. À direita da página, preencha a caixa *Register for an Activation Code*; não se esqueça de usar um endereço de e-mail válido. Em seguida, clique no botão *Download*.
5. Na nova página, baixe o pacote `Nessus-x.y.z-debian6_amd64.deb` (ajuste os valores de `x.y.z` para a versão exibida pela página). Essa versão também é indicada para o Kali Linux AMD64, que é a distribuição que estamos usando na máquina *KaliLinux-G*. Concorde com o termo de licença, e salve o pacote `.deb` — não o instale ainda.
6. No seu endereço de e-mail, cheque por uma nova mensagem com o título *Tenable Nessus Home Activation Code*. Após o cabeçalho **Activating Your Nessus Home Subscription**, o código de 20 caracteres para ativação do seu *scanner* será informado. Guarde este código para uso futuro.
7. Agora sim, vamos instalar o Nessus. O arquivo provavelmente foi baixado para a pasta `/root/Downloads`, como se segue:

```
# pwd  
/root/Downloads
```

```
# ls  
Nessus-7.1.3-debian6_amd64.deb
```

Instale-o usando o comando **dpkg**:

```
# dpkg -i Nessus-7.1.3-debian6_amd64.deb
```

```
Selecting previously unselected package nessus.  
(Reading database ... 356069 files and directories currently installed.)  
Preparing to unpack Nessus-7.1.3-debian6_amd64.deb ...  
Unpacking nessus (7.1.3) ...  
Setting up nessus (7.1.3) ...  
Unpacking Nessus Core Components...  
  
- You can start Nessus by typing /etc/init.d/nessusd start  
- Then go to https://kali:8834/ to configure your scanner  
  
Processing triggers for systemd (238-4) ...
```

Siga as instruções de instalação, e inicie o Nessus com o comando:

```
# /etc/init.d/nessusd start
```

8. Abra o navegador Firefox e acesse a URL <https://127.0.0.1:8834/> (se preferir, acesse de sua máquina física no endereço <https://172.16.0.30:8834>) para entrar na console administrativa do Nessus. Adicione uma exceção de segurança para o certificado HTTPS auto-assinado do Nessus, e prossiga.

Na tela de criação de usuário, informe o *username* **admin** e senha **rnpsr**, e clique em *Continue*.

Na tela subsequente, mantenha o *Scanner Type* em *Home, Professional or Manager*, e no campo *Activation Code* informe o código recebido por e-mail no passo (5) desta atividade. Clique em *Continue* e aguarde a inicialização do Nessus (esse passo pode demorar, seja paciente).

9. Ao final do processo, você terá acesso à console principal do Nessus, como mostrado na imagem abaixo.

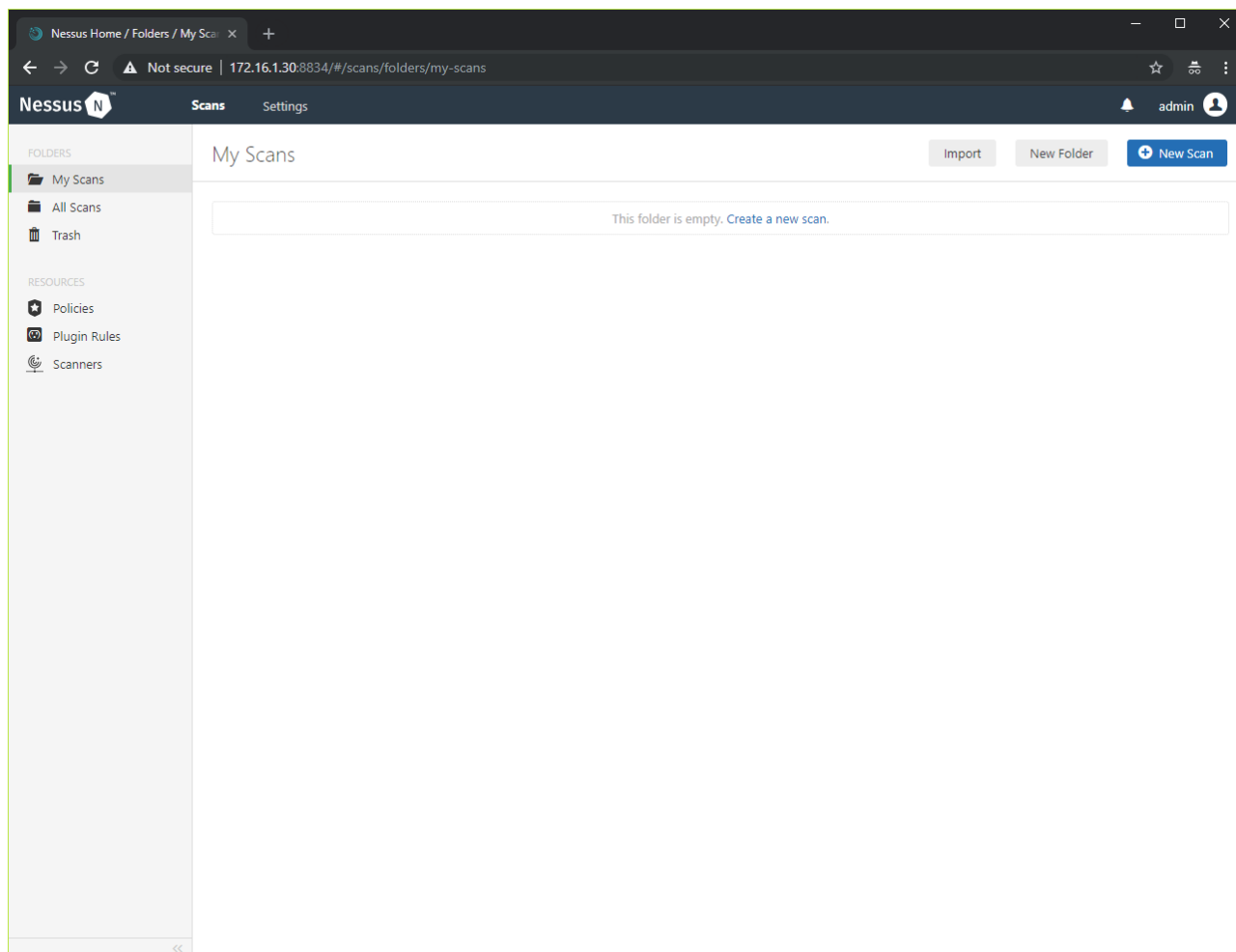


Figura 1. Console do Nessus

2) Realizando um *scan* em SO Linux



Esta atividade será realizada nas máquinas virtuais *KaliLinux-G* e *LinServer-G*.

Vamos realizar um *scan* na máquina *LinServer-G*, verificar as vulnerabilidades identificadas e tentar corrigi-las através da atualização do sistema. Antes de começar, verifique que a máquina *LinServer-G* está ligada e acessível.

1. Na console principal do Nessus, clique em *Create a new scan*. Na tela seguinte, selecione o *template Basic Network Scan*.
2. Em *Settings > General*, configure:
 - *Name*: *LinServer-G*
 - *Description*: *Scan da máquina LinServer-G*
 - *Targets*: *172.16.G.10/32*
3. Em *Credentials > SSH*, configure:
 - *Authentication method*: *password*
 - *Username*: *aluno*

- Password (unsafe!): **rnpesr**
- Elevate privileges with: **su**
- su login: **root**
- Escalation password: **rnpesr**
- Location of su (directory): **/bin**

4. Clique em *Save*. Na tela seguinte, clique no ícone *Launch* (que parece um pequeno *play*) na parte à direita da tela. O *scan* será iniciado, como mostrado abaixo.

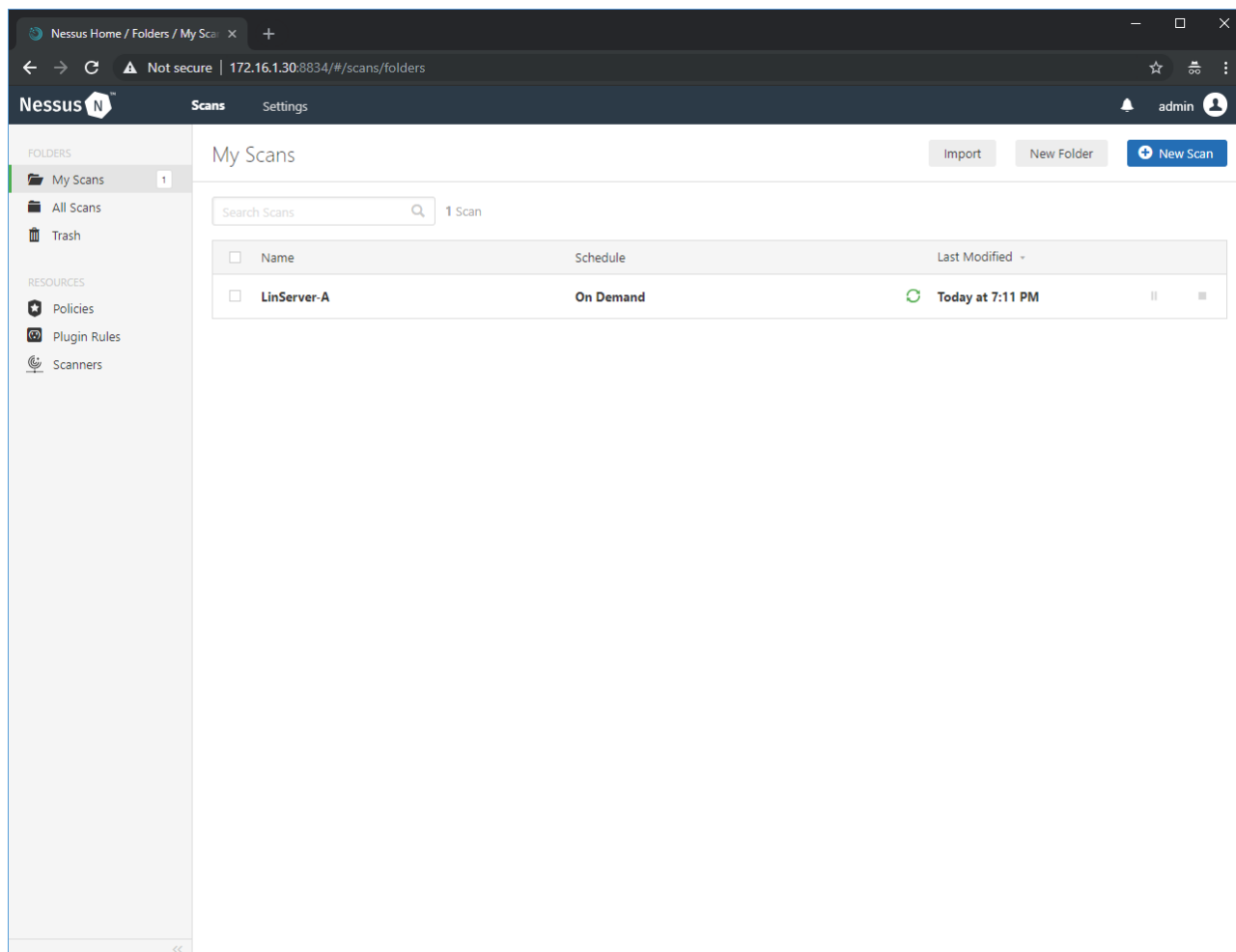


Figura 2. Scan inicial do LinServer-G no Nessus

Aguarde o final do *scan*, e confira o resultado. Se quiser acompanhar o *scan* enquanto ele é realizado, clique na linha para expandi-la.

5. Após a conclusão do *scan*, cheque a página de resultados, como mostrado abaixo.

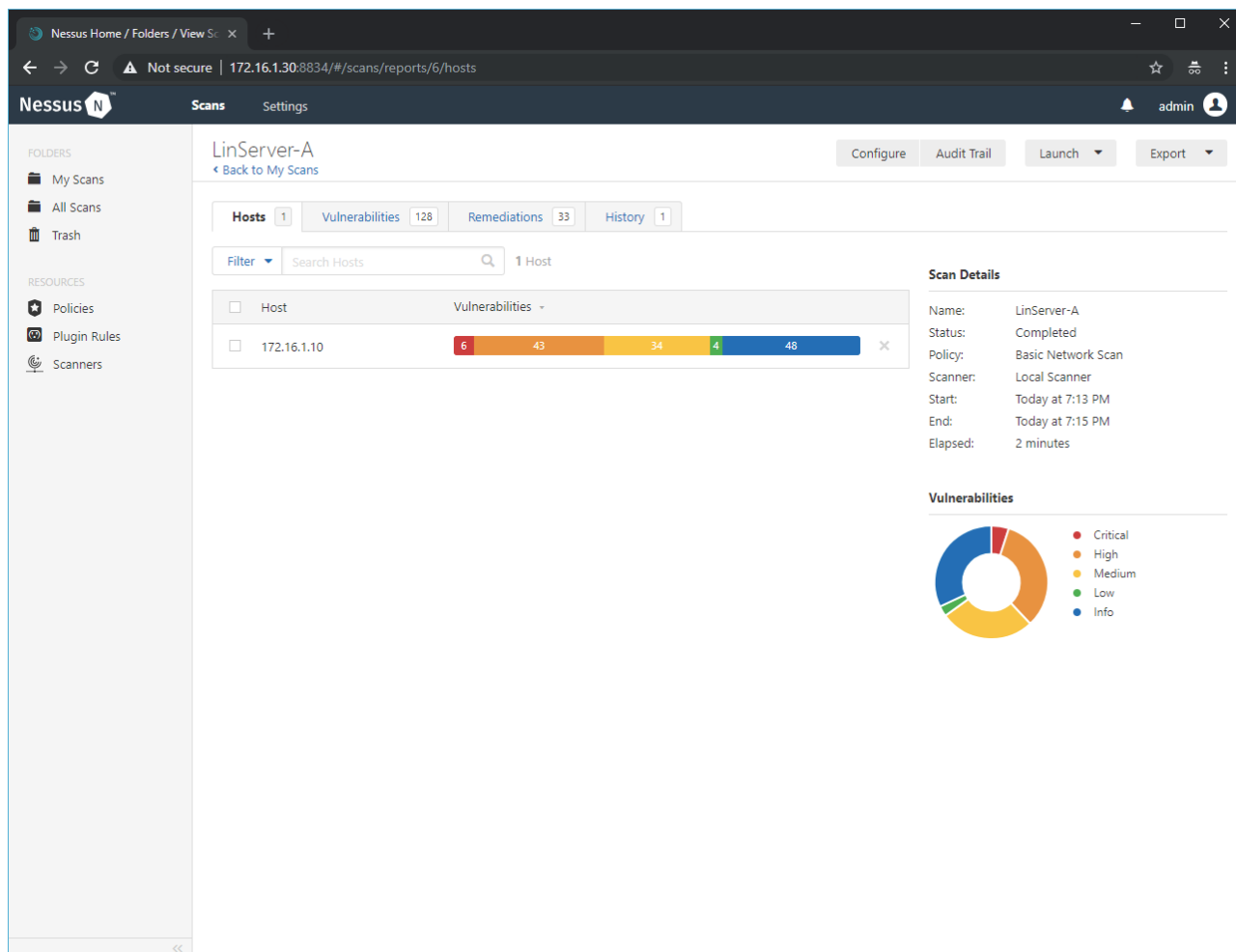


Figura 3. Primeiro scan do LinServer-G no Nessus

Veja que há um grande número de vulnerabilidades identificadas: 6 críticas, 43 de alto impacto, 34 de médio impacto, 4 de baixo impacto e 48 de cunho informativo. Entre na aba *Vulnerabilities* e explore algumas dessas vulnerabilidades—por exemplo, confira abaixo a vulnerabilidade DSA-3481-1, referente à *glibc*:

Nessus Home / Folders / View Scans | 172.16.1.30:8834/#/scans/reports/6/vulnerabilities/88768

Nessus | Scans | Settings | admin

FOLDERS

- My Scans
- All Scans
- Trash

RESOURCES

- Policies
- Plugin Rules
- Scanners

CRITICAL Debian DSA-3481-1 : glibc - security update

Description

Several vulnerabilities have been fixed in the GNU C Library, glibc.

The first vulnerability listed below is considered to have critical impact.

- CVE-2015-7547 The Google Security Team and Red Hat discovered that the glibc host name resolver function, getaddrinfo, when processing AF_UNSPEC queries (for dual A/AAAA lookups), could mismanage its internal buffers, leading to a stack-based buffer overflow and arbitrary code execution. This vulnerability affects most applications which perform host name resolution using getaddrinfo, including system services.
- CVE-2015-8776 Adam Nielsen discovered that if an invalid separated time value is passed to strftime, the strftime function could crash or leak information. Applications normally pass only valid time information to strftime; no affected applications are known.
- CVE-2015-8778 Szabolcs Nagy reported that the rarely-used hcreate and hcreate_r functions did not check the size argument properly, leading to a crash (denial of service) for certain arguments. No impacted applications are known at this time.
- CVE-2015-8779 The catopen function contains several unbound stack allocations (stack overflows), causing it to crash the process (denial of service). No applications where this issue has a security impact are currently known.

While it is only necessary to ensure that all processes are not using the old glibc anymore, it is recommended to reboot the machines after applying the security upgrade.

Solution

Upgrade the glibc packages.

For the stable distribution (jessie), these problems have been fixed in version 2.19-18+deb8u3.

See Also

- <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=812441>
- <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=812445>
- <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=812455>
- <https://security-tracker.debian.org/tracker/CVE-2015-7547>
- <https://security-tracker.debian.org/tracker/CVE-2015-8776>
- <https://security-tracker.debian.org/tracker/CVE-2015-8778>
- <https://security-tracker.debian.org/tracker/CVE-2015-8779>
- <https://packages.debian.org/source/jessie/glibc>
- <https://www.debian.org/security/2016/dsa-3481>

Plugin Details

Severity: Critical
ID: 88768
Version: \$Revision: 2.11 \$
Type: local
Family: Debian Local Security Checks
Published: February 17, 2016
Modified: January 27, 2017

Risk Information

Risk Factor: Critical
CVSS Base Score: 10.0
CVSS Vector: CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C
IAVM Severity: I

Vulnerability Information

CPE: cpe:/o:debian:debian_linux:8.0
Patch Pub Date: February 16, 2016
In the news: true

Reference Information

TRA: TRA-2017-08
DSA: 3481
IAVA: 2016-A-0053
CVE: CVE-2015-7547, CVE-2015-8776, CVE-2015-8778, CVE-2015-8779

Figura 4. Vulnerabilidade crítica da glibc

O Nessus apresenta várias informações úteis, como a natureza da vulnerabilidade, quais CVEs (*Common Vulnerabilities and Exposures*) são relevantes, e quais são as soluções mais indicadas. Do ponto de vista de gestão de riscos e vulnerabilidades em um parque com um grande número de máquinas instaladas, essas informações são importantíssimas.

- Vamos tentar corrigir algumas (ou, idealmente, todas) dessas vulnerabilidades. Entre na máquina *LinServer-G* e faça uma atualização completa do sistema. Em seguida, reinicie a VM.

```
# hostname
LinServer-A
```

```
# apt-get update
```

```
# apt-get dist-upgrade -y
```

```
# reboot
```

- De volta à console do Nessus, rode novamente o *scan* criado nos passos [1-3]. Ao final, confira

seus resultados:

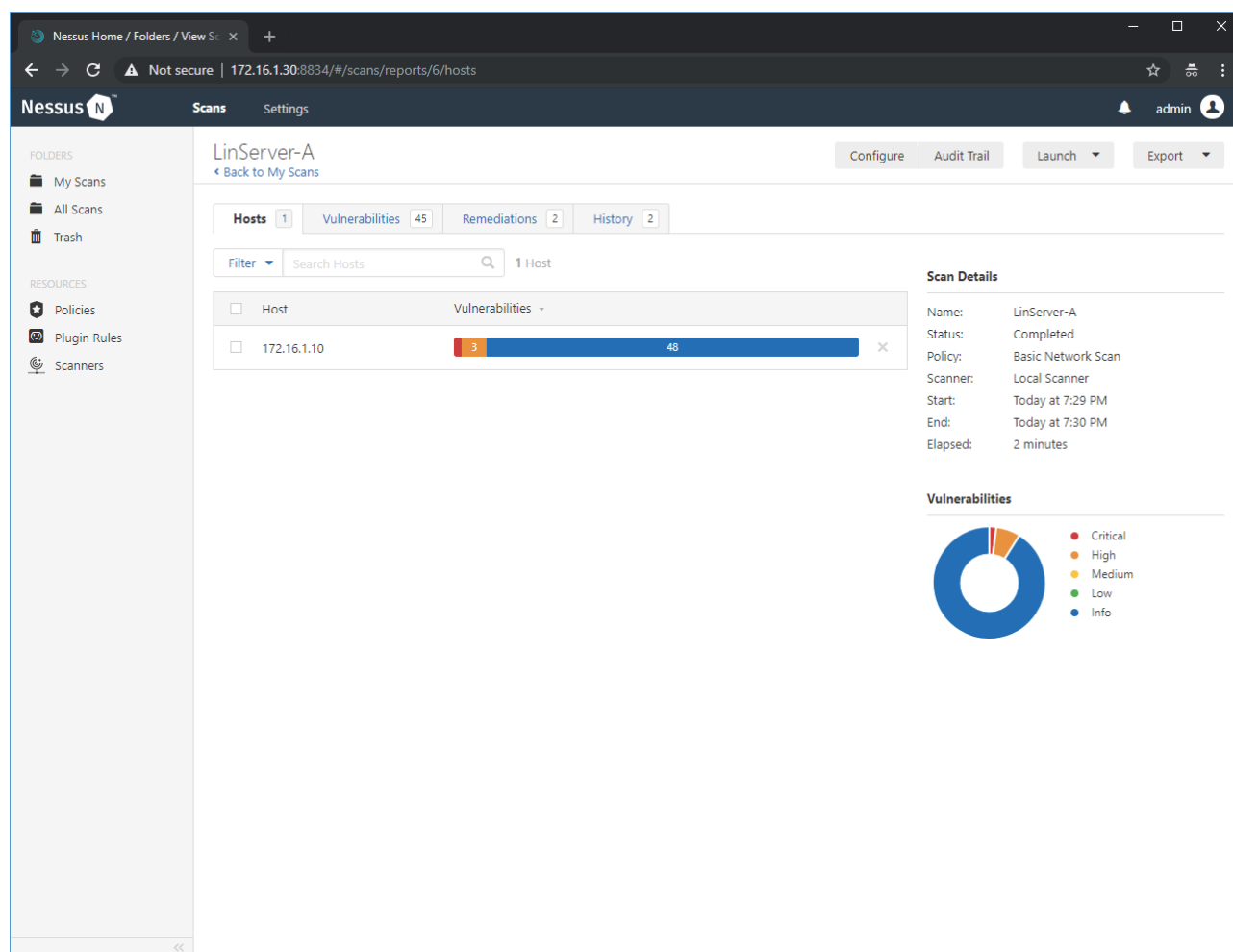


Figura 5. Scan do LinServer-G após atualização

Temos uma melhora notável: apenas 1 vulnerabilidade crítica e 3 de alto impacto foram identificadas, um cenário muito menos preocupante que o que tínhamos anteriormente.

Agora, cabe ao analista de segurança analisar cuidadosamente cada uma dessas 4 vulnerabilidades remanescentes, e determinar qual o melhor caminho a tomar para mitigá-las. Certamente, um trabalho muito mais fácil e exequível do que o que tínhamos à nossa frente antes da atualização do sistema.

3) Realizando um scan em SO Windows



Esta atividade será realizada nas máquinas virtuais *KaliLinux-G* e *WinServer-G*.

Vamos agora realizar um scan na máquina *WinServer-G*, verificar as vulnerabilidades identificadas e tentar corrigi-las via atualizações e configurações de *hardening*. Antes de começar, verifique que a máquina *WinServer-G* está ligada e acessível.

1. Na console principal do Nessus, clique em *Create a new scan*. Na tela seguinte, selecione o *template Basic Network Scan*.
2. Em *Settings > General*, configure:

- *Name:* WinServer-G
- *Description:* Scan da máquina WinServer-G
- *Targets:* 172.16.G.20/32

3. Em *Credentials* > *Windows*, configure:

- *Authentication method:* Password
- *Username:* Administrator
- *Password:* rnpesr
- *Domain:* mantenha vazio

4. Clique em *Save*. Na tela seguinte, clique no ícone *Launch* (que parece um pequeno *play*) na parte à direita da tela. O *scan* será iniciado, como anteriormente. Após a conclusão do *scan*, cheque a página de resultados, como mostrado abaixo.

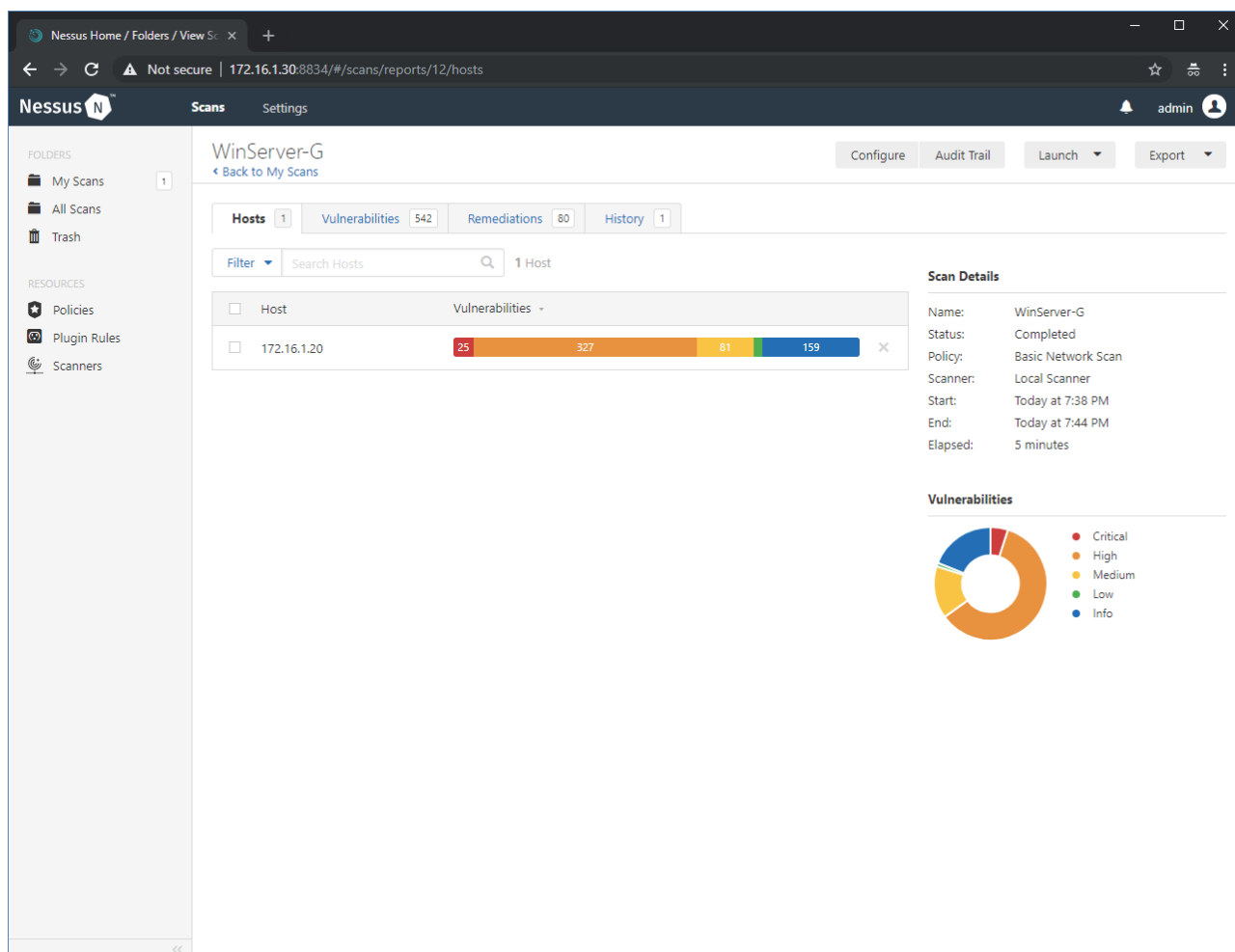


Figura 6. Primeiro scan do WinServer-G no Nessus

Veja que há um enorme número de vulnerabilidades identificadas: 25 críticas, 327 de alto impacto, 81 de médio impacto, 7 de baixo impacto e 159 de cunho informativo. Entre na aba *Vulnerabilities* e explore algumas dessas vulnerabilidades.

5. Vamos tentar corrigir algumas dessas vulnerabilidades. Entre na máquina *WinServer-G* e faça o download da ferramenta *Microsoft Baseline Security Analyzer*, em idioma inglês para máquinas x86 (disponível em <https://www.microsoft.com/en-us/download/details.aspx?id=7558>). Se

preferir, faça o download na sua máquina física e copie o instalador através da pasta compartilhada pelo Virtualbox.

Na instalação do MBSA, aceite todas as opções padrão do instalador. Em seguida, inicie a ferramenta e selecione a opção *Scan a computer*. Não altere nenhuma das opções padrão e clique em *Start Scan*. O scan será iniciado, como mostrado abaixo:

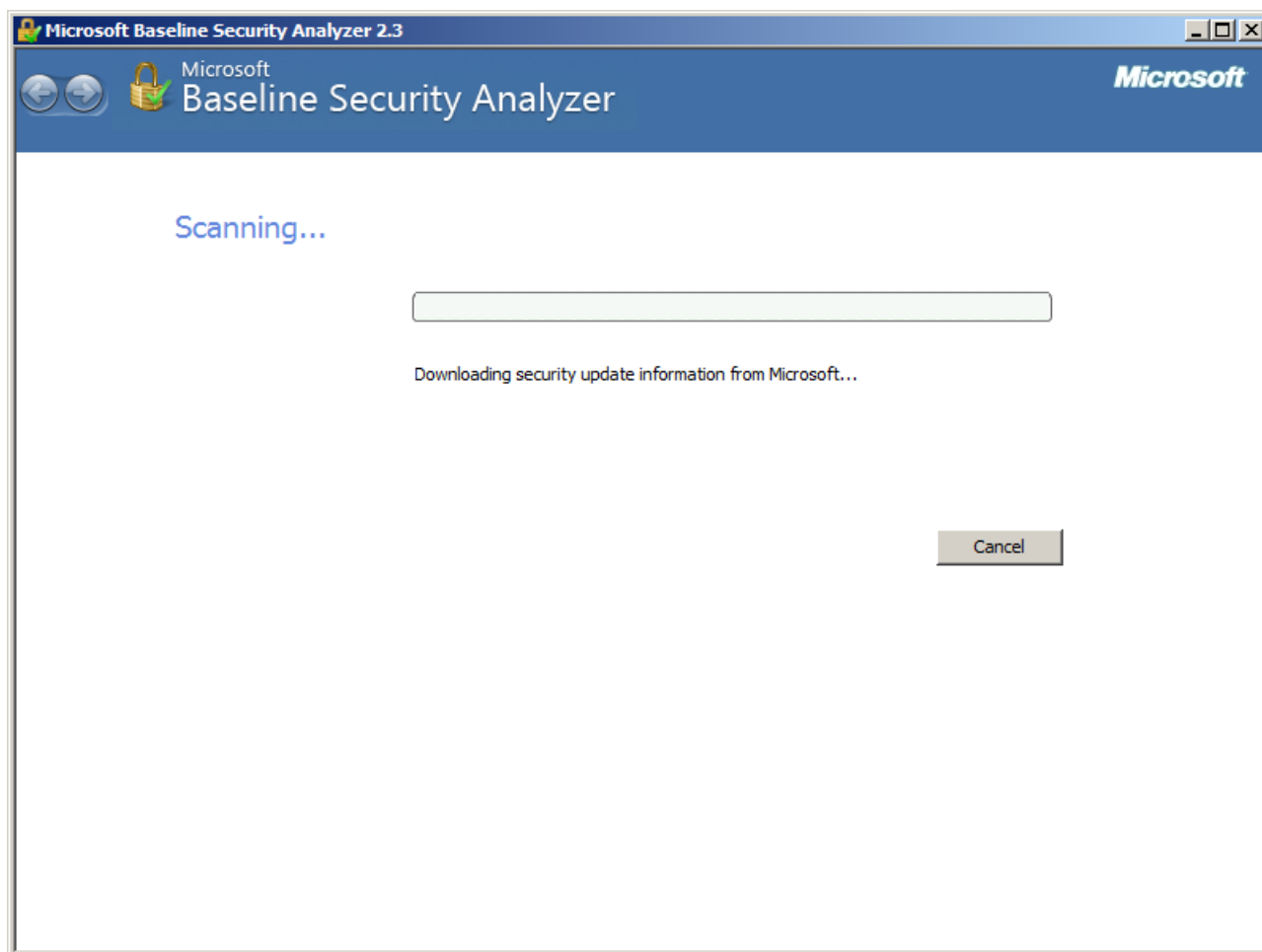


Figura 7. Scan do MBSA na máquina WinServer-G

Após o final do scan, vários apontamentos serão indicados pelo MBSA, como se segue:

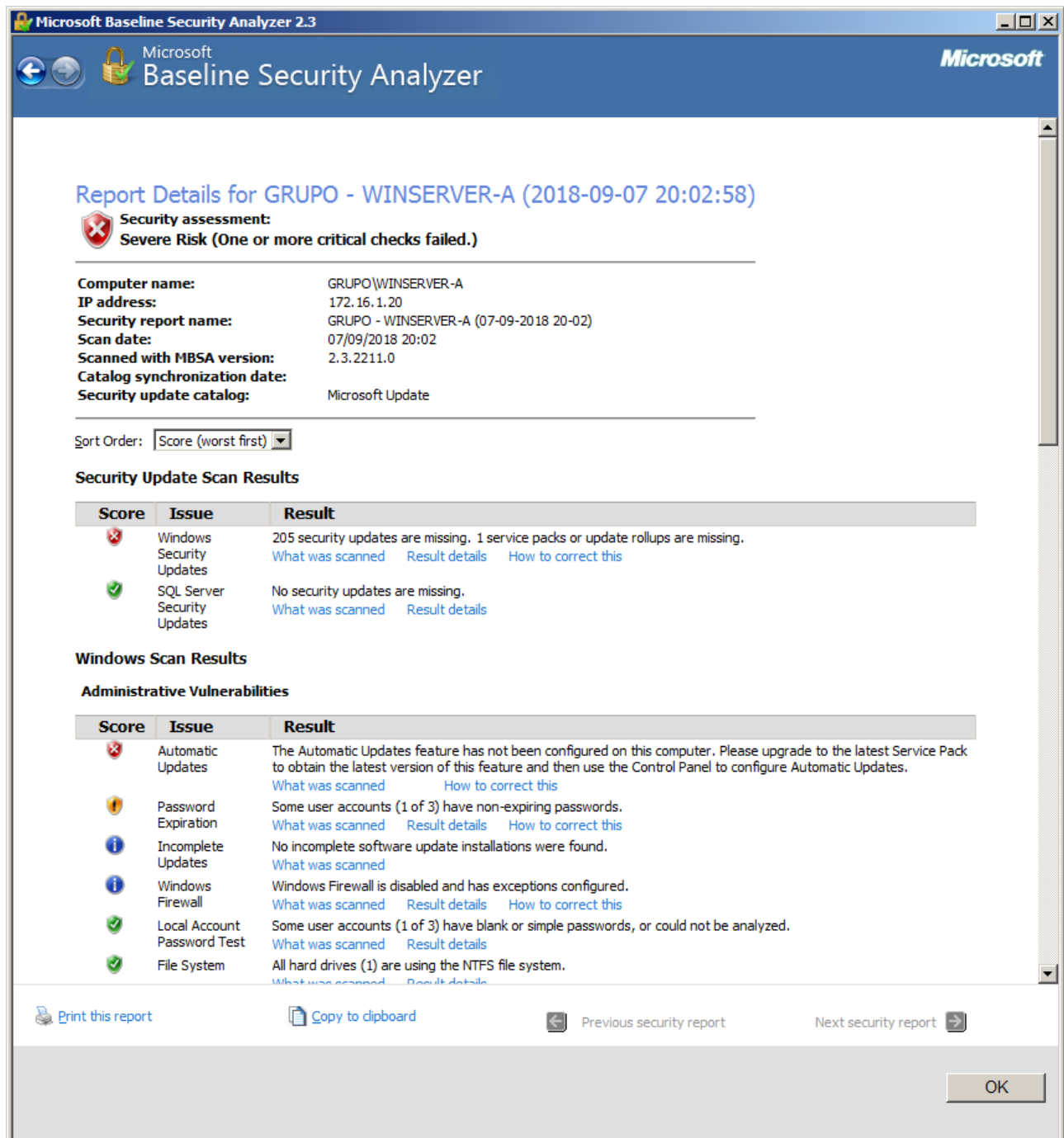


Figura 8. Resultados do scan do MBSA na máquina WinServer-G

Desses, o mais preocupante é de longe o grande número de atualizações de segurança pendentes: 205. Ainda há alertas quanto à falta de atualizações automáticas, expiração de senhas de usuários e situação do *Windows Firewall*.

- Seguindo as recomendações do MBSA, ative as atualizações automáticas e faça a atualização completa da máquina *WinServer-G*. Como esperado, esse passo pode demorar um pouco, então seja paciente.

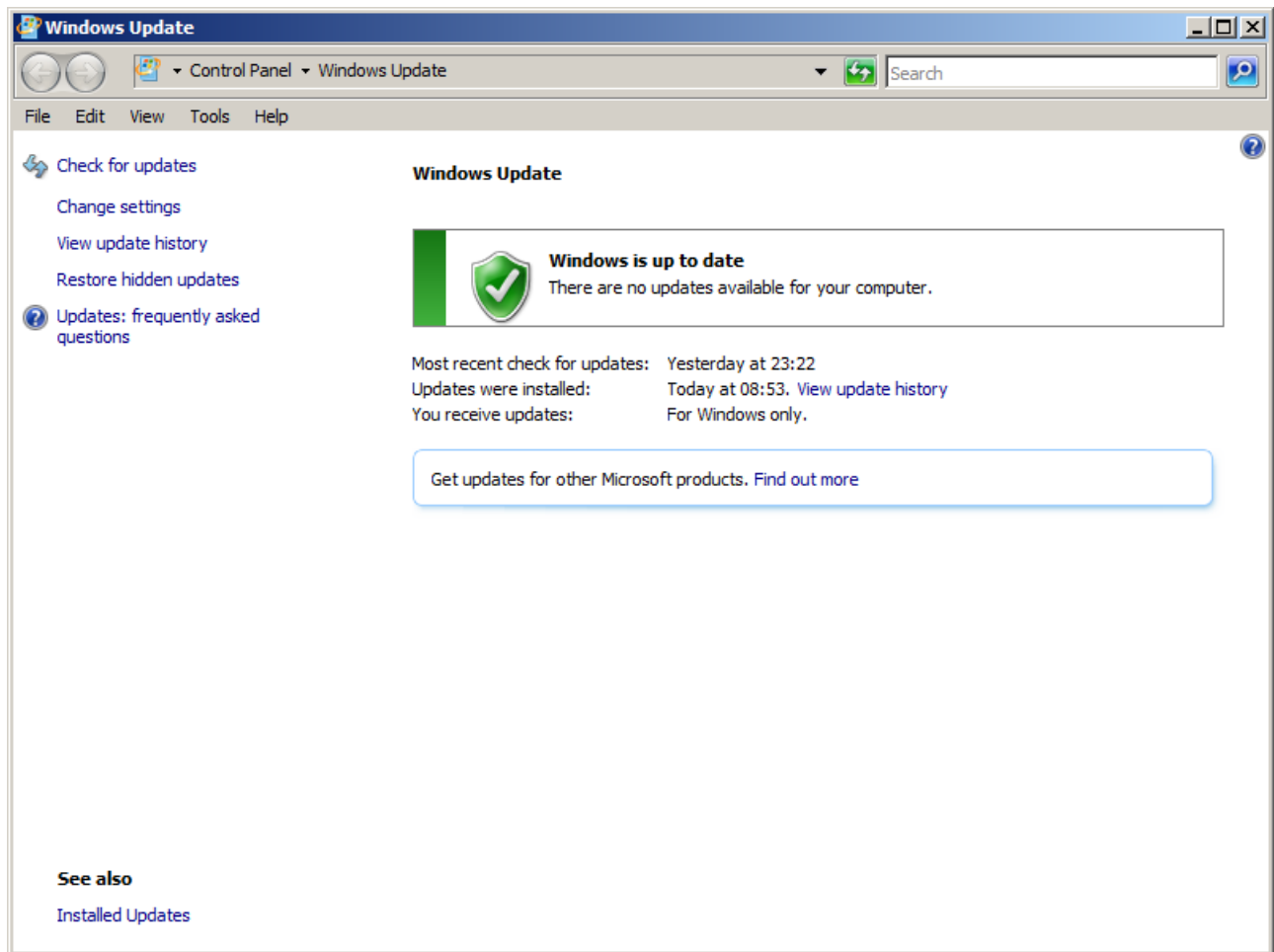


Figura 9. WinServer-G atualizado

Após o final do processo, a tela do *Windows Update* deve mostrar a mensagem acima.

7. Rode novamente o *scan* do Nessus na máquina *WinServer-G* e verifique os resultados.

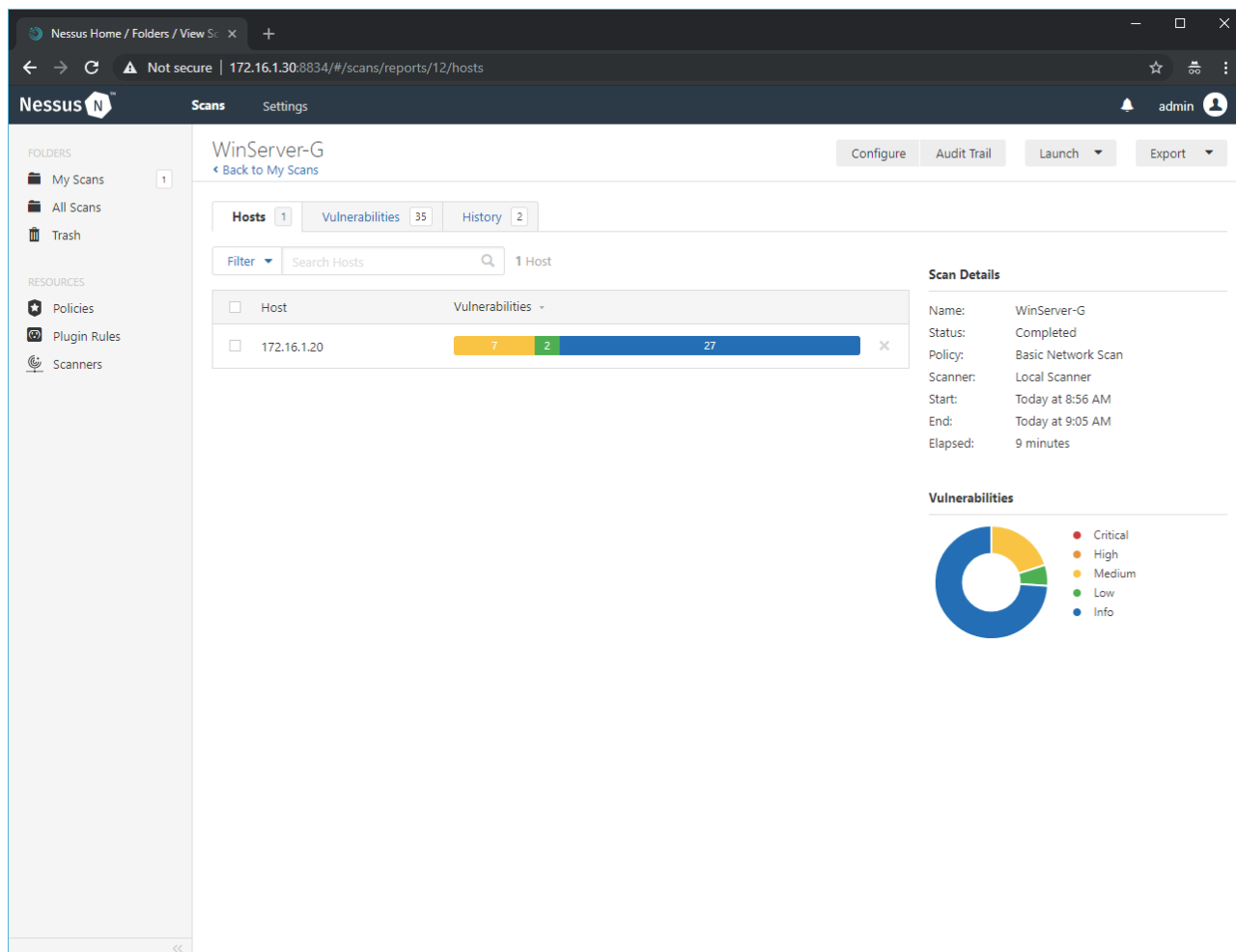


Figura 10. Scan final do WinServer-G no Nessus

A diferença para o panorama anterior é significativa: agora temos apenas 7 vulnerabilidades de médio impacto, 2 de baixo impacto e 27 de cunho informativo. De fato, as recomendações do MBSA e as atualizações de sistema fizeram uma diferença importante na segurança do sistema.

4) Efeitos de firewall e IDS em um scan



Esta atividade será realizada nas máquinas virtuais *KaliLinux-G* e *FWGW1-G*.

Vamos agora realizar um *scan* na máquina *FWGW1-G*. Lembre-se, no entanto, que além de o firewall interno (especificamente, da *chain* INPUT da tabela *filter*) ser bastante restritivo, temos o Snort alertando sobre comportamentos anômalos na rede. Qual será o efeito desses elementos em um *scan* do Nessus?

1. Na console principal do Nessus, clique em *Create a new scan*. Na tela seguinte, selecione o *template Basic Network Scan*.
2. Em *Settings > General*, configure:
 - *Name*: *FWGW1-G*
 - *Description*: Scan da máquina *FWGW1-G*
 - *Targets*: 172.16.G.1/32

3. Não iremos adicionar login via `ssh` para este *scan*, por dois motivos: primeiro, queremos testar o impacto das proteções de rede que empregamos na efetividade do *scan* e, segundo, porque não há regra que permita logins `ssh` oriundos da rede 172.16.G.0/24.
4. Clique em *Save*. Antes de iniciar o *scan* propriamente dito, logue na máquina *FWGW1-G* como usuário `root`. Queremos monitorar os logs do Snort durante o *scan*, para ver os alertas levantados pelo IDS. Só temos um problema —no momento, o Snort está monitorando a interface `eth0`:

```
# hostname  
FWGW1-A
```

```
# cat /etc/systemd/system/snort.service | grep ExecStart  
ExecStart=/usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i  
eth0 -D
```

No entanto, a máquina *KaliLinux-G* está conectada à rede DMZ, assim como a interface `eth1` do firewall. Pare a execução do Snort e inicie-o manualmente na interface `eth1`; em seguida, monitore seus alertas no arquivo `/var/log/snort/alert`:

```
# systemctl stop snort
```

```
# /usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth1 -D
```

```
# tail -f -n0 /var/log/snort/alert
```

Agora sim, clique no ícone *Launch* (que parece um pequeno *play*) na parte à direita da tela. O *scan* será iniciado, como anteriormente. Após a conclusão do *scan*, cheque a página de resultados, como mostrado abaixo.

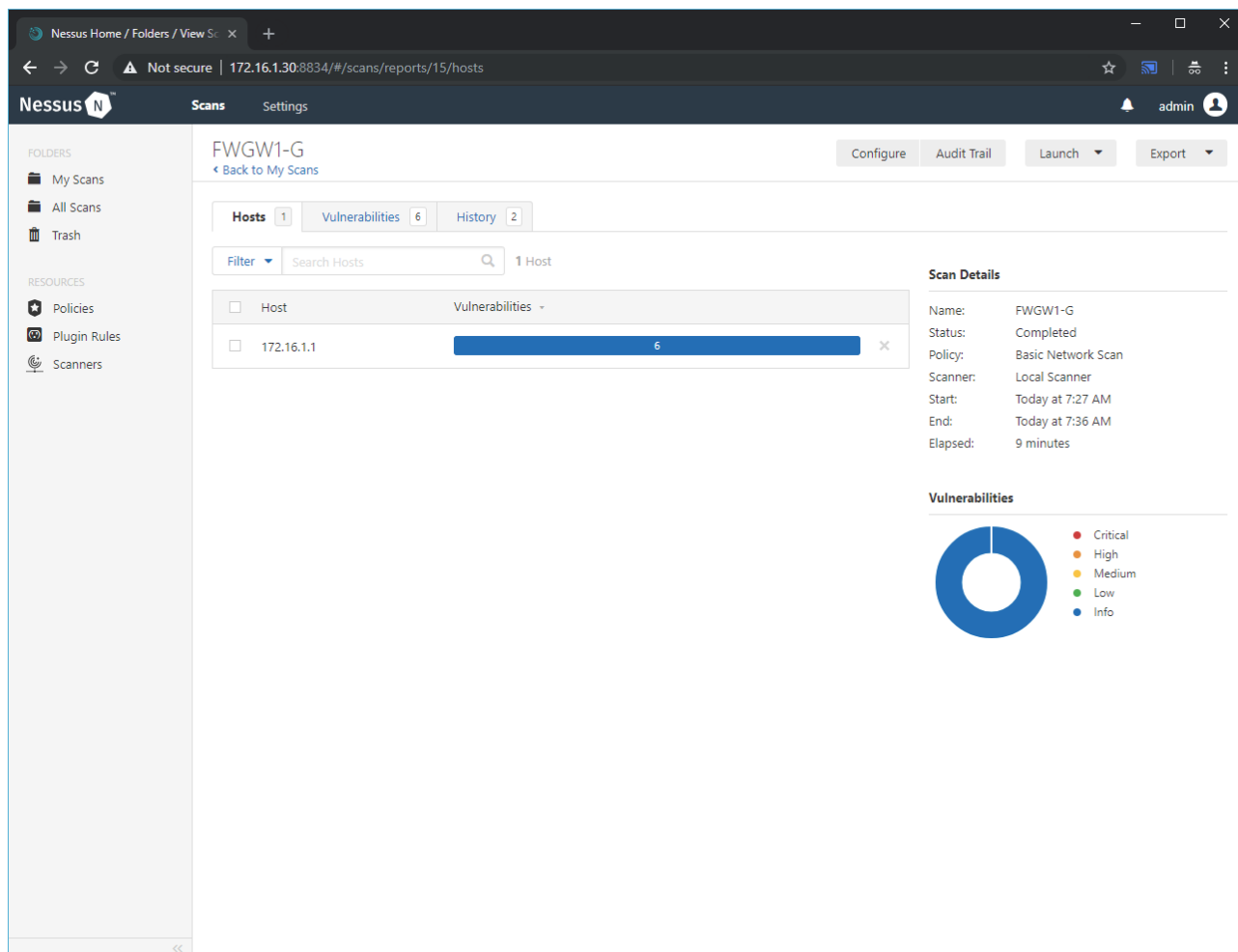


Figura 11. Primeiro scan do FWGW1-G no Nessus

Um resultado impressionante: apenas 6 vulnerabilidades informativas, e nenhum alerta levantado pelo Snort. Mas, será mesmo?

5. Limpe as configurações de firewall da máquina *FWGW1-G*, permitindo todo tipo de conexão externa. Em seguida, reinicie o monitoramento do arquivo de log do Snort e rode o *scan* novamente.

```
# iptables -P INPUT ACCEPT
```

```
# iptables -P FORWARD ACCEPT
```

```
# iptables -F
```

```
# iptables -L -vn
Chain INPUT (policy ACCEPT 55 packets, 6971 bytes)
  pkts bytes target    prot opt in     out     source                   destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source                   destination

Chain OUTPUT (policy ACCEPT 20 packets, 1568 bytes)
  pkts bytes target    prot opt in     out     source                   destination
```

```
# tail -f -n0 /var/log/snort/alert
```

Feito isso, clique novamente no botão *Launch* para iniciar um novo *scan*. De imediato, os logs do Snort começam a acusar tráfego suspeito:

```
[**] [129:15:1] Reset outside window [**]
[**] [129:15:1] Reset outside window [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
[Classification: Potentially Bad Traffic] [Priority: 2]
09/08-06:43:38.018485 172.16.1.30:55498 -> 172.16.1.1:22
09/08-06:43:38.018485 172.16.1.30:55498 -> 172.16.1.1:22
TCP TTL:255 TOS:0x0 ID:2746 IpLen:20 DgmLen:40
TCP TTL:255 TOS:0x0 ID:2746 IpLen:20 DgmLen:40
*****R** Seq: 0x4766526D Ack: 0x0 Win: 0x200 TcpLen: 20
*****R** Seq: 0x4766526D Ack: 0x0 Win: 0x200 TcpLen: 20

[**] [129:15:1] Reset outside window [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
09/08-06:43:39.087546 172.16.1.30:55498 -> 172.16.1.1:22
TCP TTL:255 TOS:0x0 ID:2746 IpLen:20 DgmLen:40
*****R** Seq: 0x4766526D Ack: 0x0 Win: 0x200 TcpLen: 20

[**] [129:15:1] Reset outside window [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
09/08-06:43:39.087546 172.16.1.30:55498 -> 172.16.1.1:22
TCP TTL:255 TOS:0x0 ID:2746 IpLen:20 DgmLen:40
*****R** Seq: 0x4766526D Ack: 0x0 Win: 0x200 TcpLen: 20

[**] [128:4:1] (spp_ssh) Protocol mismatch [**]
[Classification: Detection of a non-standard protocol or event] [Priority: 2]
09/08-06:44:21.814817 172.16.1.30:55520 -> 172.16.1.1:22
TCP TTL:64 TOS:0x0 ID:64890 IpLen:20 DgmLen:576 DF
***AP*** Seq: 0x1196C581 Ack: 0x0 Win: 0x0 TcpLen: 32
```

Após o final do *scan*, temos o seguinte resultado:

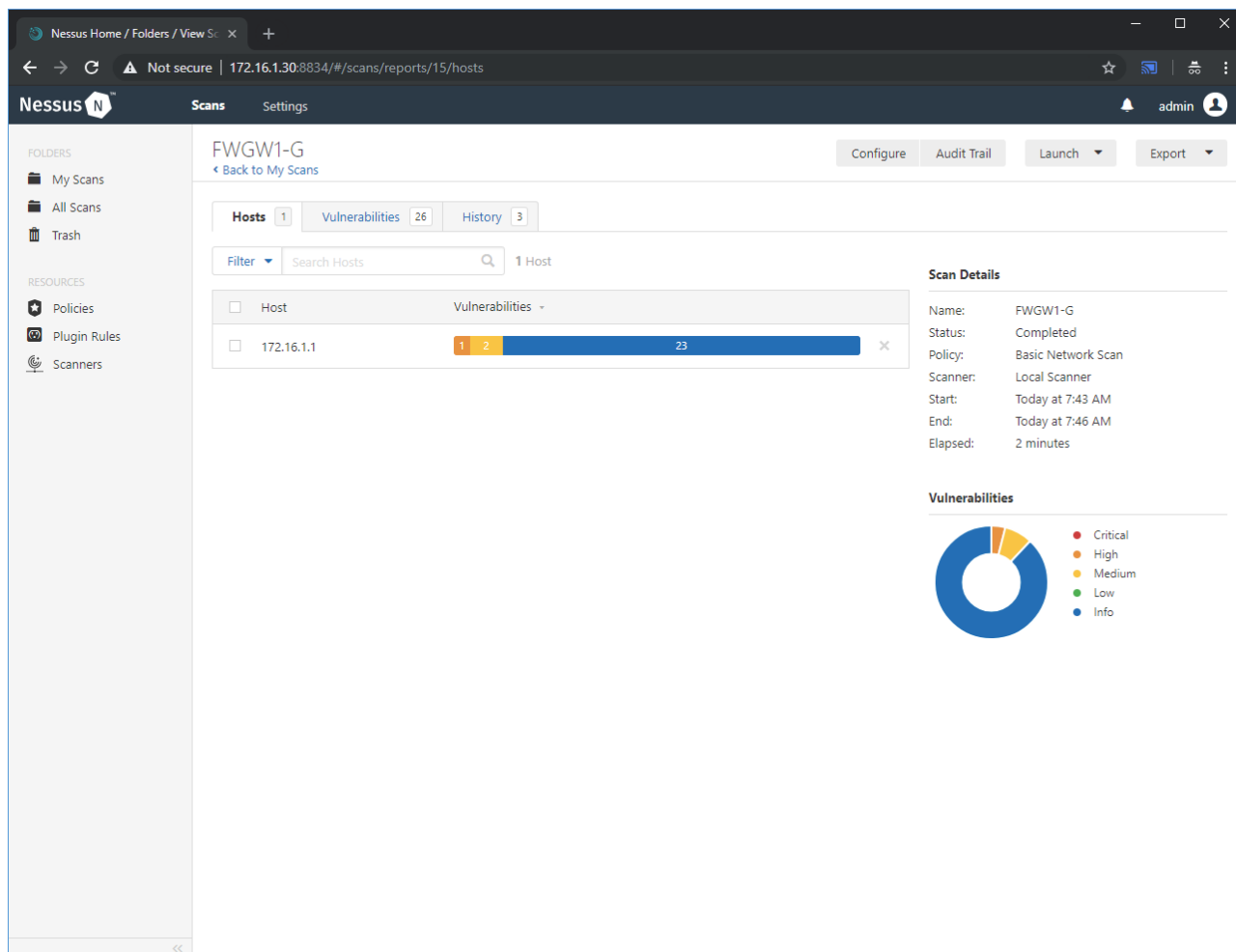


Figura 12. Segundo scan do FWGW1-G no Nessus

Agora temos uma vulnerabilidade de alto impacto, duas de médio impacto e 23 informativas. Talvez o servidor não esteja tão seguro quando imaginávamos... vamos tentar ir mais a fundo.

6. Dentro do scan do FWGW1-G, clique em *Configure*. Em *Credentials > SSH*, configure:

- *Authentication method*: password
- *Username*: aluno
- *Password (unsafe!)*: rnpesr
- *Elevate privileges with*: su
- *su login*: root
- *Escalation password*: rnpesr
- *Location of su (directory)*: /bin

Clique em *Save*, e rode o scan uma terceira vez. Perceba que os logs do Snort continuam alertando sobre tráfego suspeito, quase que exclusivamente direcionado à porta 22:

```
[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**]  
[Classification: Potentially Bad Traffic] [Priority: 2]  
09/08-06:52:33.667651 172.16.1.30:55974 -> 172.16.1.1:22  
TCP TTL:64 TOS:0x0 ID:45684 IpLen:20 DgmLen:104 DF  
***AP*** Seq: 0xFE2FB6F5 Ack: 0x9A8250FC Win: 0x102 TcpLen: 32  
TCP Options (3) => NOP NOP TS: 4021996676 13125828
```

```
[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**]  
[Classification: Potentially Bad Traffic] [Priority: 2]  
09/08-06:52:33.671810 172.16.1.30:55974 -> 172.16.1.1:22  
TCP TTL:64 TOS:0x0 ID:45689 IpLen:20 DgmLen:104 DF  
***AP*** Seq: 0xFE2FB7C5 Ack: 0x9A825278 Win: 0x102 TcpLen: 32  
TCP Options (3) => NOP NOP TS: 4021996680 13125829
```

```
[**] [129:12:1] Consecutive TCP small segments exceeding threshold [**]  
[Classification: Potentially Bad Traffic] [Priority: 2]  
09/08-06:52:33.679830 172.16.1.30:55972 -> 172.16.1.1:22  
TCP TTL:64 TOS:0x0 ID:31597 IpLen:20 DgmLen:104 DF  
***AP*** Seq: 0xB5706A01 Ack: 0x397D0971 Win: 0x111 TcpLen: 32  
TCP Options (3) => NOP NOP TS: 4021996688 13125831
```

Isso se deve ao fato de que a máquina *FWGW1-G* possui pouquíssimos serviços escutando externamente:

```
# netstat -tunlp | grep -v '127.0.0.1' | grep -v '^tcp6\|^udp6'
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
531/sshd
udp        0      0 0.0.0.0:68              0.0.0.0:*
417/dhclient
udp        0      0 10.8.1.1:123            0.0.0.0:*
576/ntpd
udp        0      0 10.1.1.1:123            0.0.0.0:*
576/ntpd
udp        0      0 172.16.1.1:123          0.0.0.0:*
576/ntpd
udp        0      0 192.168.29.103:123      0.0.0.0:*
576/ntpd
udp        0      0 0.0.0.0:123             0.0.0.0:*
576/ntpd
udp        0      0 0.0.0.0:161             0.0.0.0:*
629/snmpd
udp        0      0 0.0.0.0:1194            0.0.0.0:*
562/openvpn
udp        0      0 0.0.0.0:55650           0.0.0.0:*
629/snmpd
udp        0      0 0.0.0.0:50561           0.0.0.0:*
417/dhclient
```

Dos serviços acima, apenas o **ssh** e o **openvpn** estão ativamente escutando por conexões externas (o **ntpd** apenas consulta o servidor de hora *LinServer-G*).

E como ficou o resultado do *scan*?

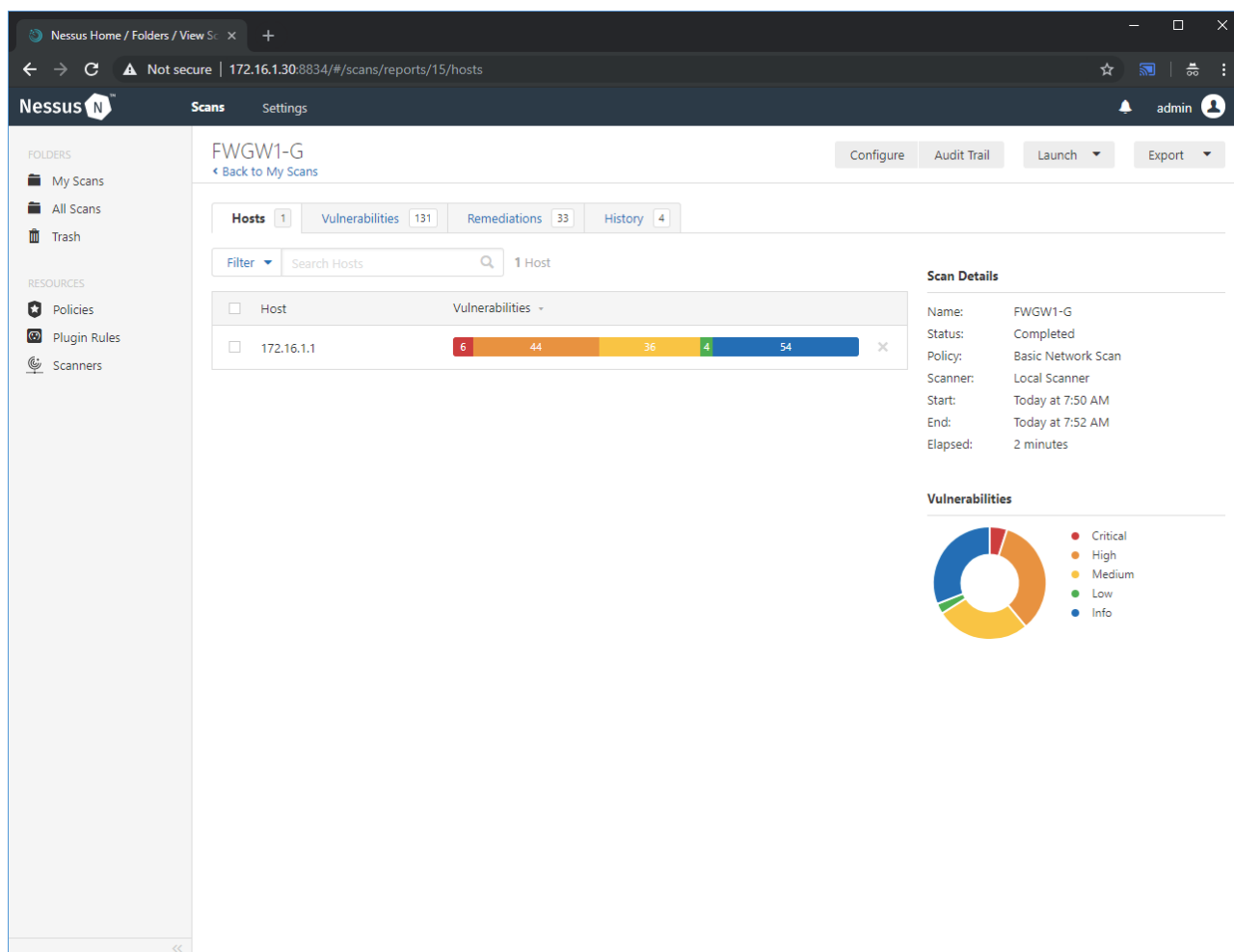


Figura 13. Scan final do FWGW1-G no Nessus

Com o login `ssh` ativado, o Nessus conseguiu, agora sim, encontrar 6 vulnerabilidades críticas, 44 de alto impacto, 36 de médio impacto, 4 de baixo impacto e 54 informativas. De fato, a segurança do servidor *FWGW1-G* está no mesmo patamar da máquina *LinServer-G* antes da sua atualização, o que seria esperado.

Esse exercício serve para visualizarmos um fato relevante: firewalls e ferramentas IPS (no caso, nosso Snort está apenas atuando como IDS, no momento) podem mascarar problemas de segurança, que ficam latentes até que um atacante descubra um método de aproveitar-se delas. Sempre que for rodar ferramentas de análise de vulnerabilidades automatizadas em sua rede, lembre-se de criar regras de liberação relevantes nos firewalls para visualizar a real situação do seu parque.

7. Atualize a máquina *FWGW1-G* e rode o *scan* novamente, nos mesmos moldes que fizemos com o *LinServer-G*. Houve melhora significativa?
8. Finalmente, recarregue as regras de firewall para seu estado original, pare o Snort e reinicie-o na interface `eth0` como usual.

```
# systemctl restart netfilter-persistent.service
```

```
# iptables -vn -L
Chain INPUT (policy DROP 53 packets, 14973 bytes)
  pkts bytes target     prot opt in     out     source            destination
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0         0.0.0.0/0
    0     0 REJECT     all  --  !lo    *       0.0.0.0/0         127.0.0.0/8
reject-with icmp-port-unreachable
   89  4720 ACCEPT     all  --  *      *       0.0.0.0/0         0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      *       10.1.1.0/24       0.0.0.0/0
tcp dpt:22 state NEW,ESTABLISHED

(...)
```

```
# ps auxwm | grep '^snort '
snort      1575  0.0 26.5 629172 545672 ?        -    06:26   0:00
/usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth1 -D
snort      -  0.0  -    -    -    -        Ssl  06:26   0:00 -
snort      -  0.0  -    -    -    -        Ssl  06:26   0:00 -
```

```
# kill 1575
```

```
# systemctl start snort
```

5) Auditoria de servidores web



Esta atividade será realizada nas máquinas virtuais *KaliLinux-G*, *LinServer-G* e *WinServer-G*.

1. Na máquina *KaliLinux-G*, execute a ferramenta **nikto** buscando por vulnerabilidades no servidor web Apache instalado na máquina *LinServer-G*.

```
# nikto -host 172.16.1.10 -C all
- Nikto v2.1.6

-----
+ Target IP:          172.16.1.10
+ Target Hostname:    172.16.1.10
+ Target Port:        80
+ Start Time:         2018-09-08 08:21:38 (GMT-3)
-----
+ Server: Apache/2.4.10 (Debian)
+ Server leaks inodes via ETags, header found with file /, fields: 0x29cd
0x5744726bfc360
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user
agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to
render the content of the site in a different fashion to the MIME type
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.12). Apache
2.0.65 (final release) and 2.2.29 are also current.
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ OSVDB-3233: /icons/README: Apache default file found.
+ 26165 requests: 0 error(s) and 7 item(s) reported on remote host
+ End Time:           2018-09-08 08:22:11 (GMT-3) (33 seconds)
-----
+ 1 host(s) tested
```

O **nikto** é um *scanner* de servidores web *open source* que faz testes profundos procurando por arquivos/programas perigosos, versões de serviço desatualizadas, bem como problemas de configuração e exposição de dados. É uma ferramenta muito poderosa para identificar problemas comuns em servidores web, e deve sempre ser considerada pelo analista de segurança em suas análises.

No caso específico da máquina *LinServer-G*, como apenas fizemos a instalação do Apache e não há nenhum website instalado, o número de vulnerabilidades encontradas é baixo, quase todas informativas.

2. Use o **nikto** para escanear o servidor web IIS instalado na máquina *WinServer-G*.


```
# nikto -host 172.16.1.20 -C all
- Nikto v2.1.6

-----
+ Target IP:          172.16.1.20
+ Target Hostname:    172.16.1.20
+ Target Port:        80
+ Start Time:         2018-09-08 08:47:07 (GMT-3)
-----

+ Server: Microsoft-IIS/7.0
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user
agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to
render the content of the site in a different fashion to the MIME type
+ Allowed HTTP Methods: OPTIONS, TRACE, GET, HEAD, POST
+ Public HTTP Methods: OPTIONS, TRACE, GET, HEAD, POST
+ /: Appears to be a default IIS 7 install.
+ 26165 requests: 0 error(s) and 6 item(s) reported on remote host
+ End Time:           2018-09-08 08:50:32 (GMT-3) (205 seconds)
-----

+ 1 host(s) tested
```

Da mesma forma que o *host LinServer-G*, a instalação do IIS na máquina *WinServer-G* é basicamente a padrão e, especialmente após a atualização do sistema que fizemos na atividade (3) desta sessão, apresenta apenas notificações informativas.