Número #17 3 a 5 de Outubro de 2013

slackware zine

www.slackwarezine.com.br

Automação simplificada com Slackware e um “computador de bolso”

Juliana Silvestre - juliana@cbpf.br
Marcelo Giovani - mgm@cbpf.br
Centro Brasileiro de Pesquisas Físicas - CBPF/MCTI

- Roteiro

* Justificativa
* Hardware utilizado
  - Histórico do RPI
  - Especificações
    .Gerais
    .BCM2835
    .Barramento
  -Processo de boot
* SlackwareARM
* Instalação
  -Ajustes iniciais
  -Processo de instalação
  -Pós-instalação
* GPIO
* Resultados
  -Hardware
  -Controle
* Conclusão
* Referências
* Agradecimentos

- Justificativa

* Monitoração:

Necessidade de se inserir um hardware em locais de difícil acesso

Sob forro de teto;
Salas sem climatização;
Dificuldade de alimentação elétrica;
Espaço físico dificultado em interior de Racks e bastidores de telecom;
Dutos de refrigeração; etc.

* Atuação:

Necessidade de promover ações corretivas à ocorrências (elétricas) ou atuação preventiva planejada

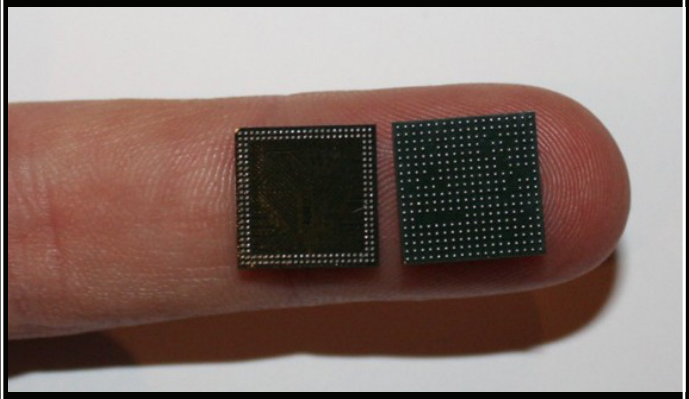
* União de “controle de sinais elétricos” + “acesso remoto via rede” (web)

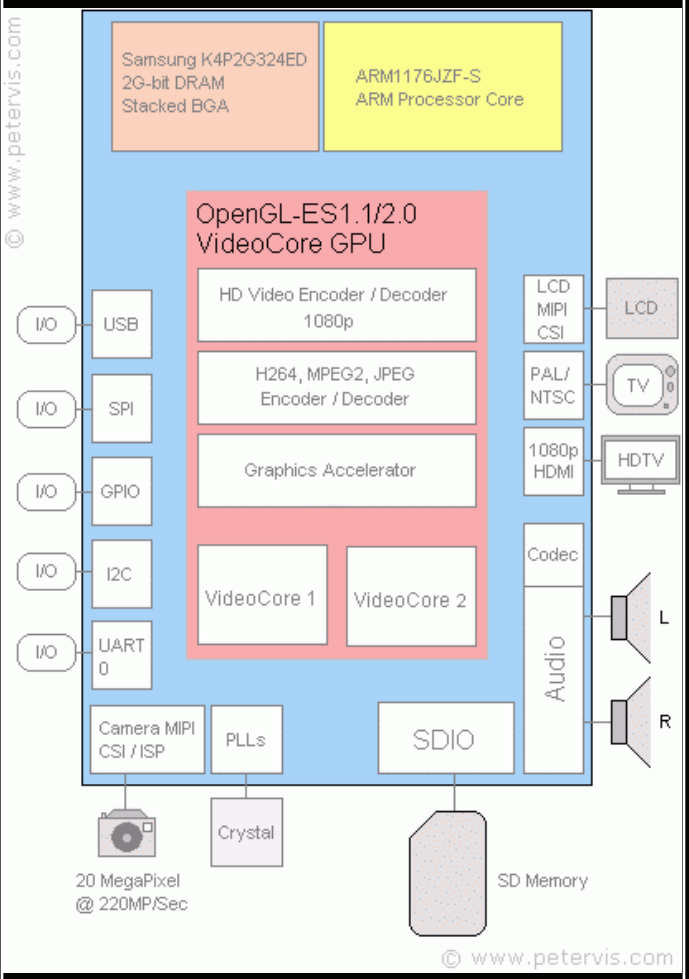
* Possibilidade de alimentação via cabo de rede (PoE)

* Possibilidade de alimentação por painéis solares / baterias

- Hardware

* RaspberryPi®:
  - Quando - 2006
  - Quem - Eben Upton, Rob Mullins, Jack Lang e Alan Mycroft
  - Onde - Reino Unido; Laboratórios de Computação da Universidade de Cambridge
* Especificações:
  - Gerais
    - System on a Chip (SoC) Broadcom BCM2835
    - ARM11 700MHz (armv6k)
    - GPU:
      - FullHD (1080p)
        Compatível com BluRay quality playback, H.264 em 40MBits/s
        Suporte 3D
        OpenGL ES2.0
        OpenCV
    - 512MB de memória (256MB anteriormente)
    - Saída de vídeo digital HDMI
    - Saída de vídeo analógica RCA (vídeo composto)
    - Placa de áudio (saída via HDMI e/ou analógica 3,5mm)
    - 2 x USB (2.0)
    - RJ45 (ethernet 10/100 FD)
    - Alimentação 5Vcc (Micro USB)
      Consumo 2,5 ~ 3,5w
* BCM2835
  - System on a Chip (SoC) Broadcom BCM2835:
    - ARM11 700MHz (armv6k)
    - GPU
    - Memória (Samsung)
    - USB (2.0)
    - GPIO
    - SPI
    - I²C
    - UART





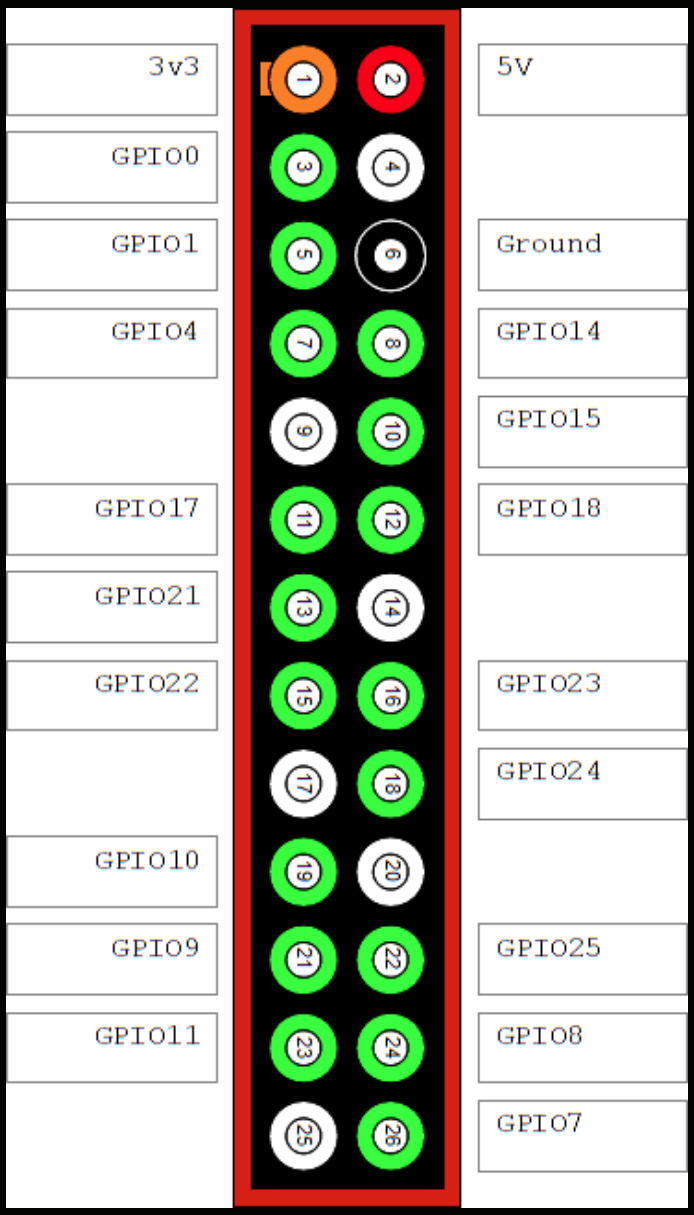
Barramento

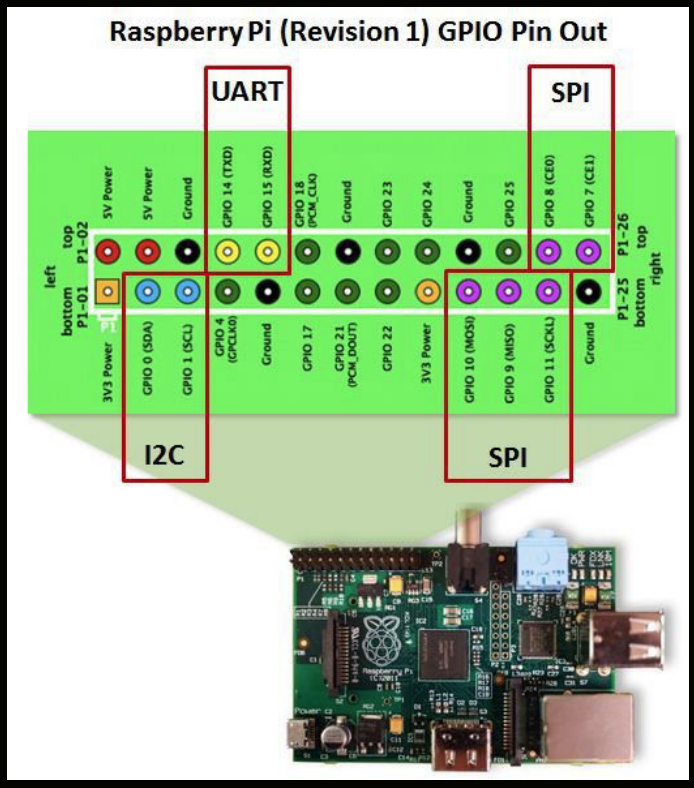
GPIO - General Purpose Input/Output (3,3Vcc)

I²C - Inter-Integrated Circuit [Philips - uso livre desde 1/out/2006]

SPI - Serial Peripheral Interface

UART serial (Universal Asynchronous Receiver/Transmitter)





* Processo de boot:

Acionamento inicial - GPU (ARM atua como co-processador)

.Execução de código interno (ROM interna)

.Busca da 1ª partição do SD, execução do “bootcode.bin” (GPU)

.Inicia memória SDRAM, busca e execução do “start.elf” (GPU)

.Auxiliado pelo “fixup.dat” e “config.txt”*** configura o hardware e inicia o kernel linux

***Config.txt:

.HDMI mode

.PAL/NTSC

.Resolução

.Vídeo memory size

.Aspect (4:3 ; 14:9 ; 16:9)

.Frequência de vídeo

.Rotação de display

.Memery freq; core freq; over_voltage...

*** http://elinux.org/RPi_config.txt

- SlackwareARM

Quando - 2002

Quem - Stuart Winter

Nome - ARMedslack

2007 - ARMedslack (slackware11.0)

2009 abril 02 - P. Volkerding, port oficial slackware ARM - Slackware ARM

2009 jun - SlackwareARM-12.2

Atual - SlackwareARM-14.0

* Suporte:

Inicialmente a RiscPC com a versão ARMedslack-11.0

SlackwareARM-12.2 - “velha” ABI

SlackwareARM-13.0 - X

SlackwareARM-13.1 - version4 + EABI

http://arm.slackware.com/introduction/

- Instalação

* Ajustes iniciais:

.Baixar uma imagem de instalação (imagem de boot)

http://www.cbpf.br/~juliana/files/raspi-slack-installer.img.gz

.Descompactar e copiar para a raiz de um cartão SD de 4GB ou mais

(dd ; ddrescue …)

.Preparar a fonte dos pacotes slackwareARM

Pendrive local (~1:50h - “A ; AP ; N”)

HTTP (~3:30h - “A ; AP ; N”)

- uma instalação de toda árvore slackware pode demorar mais de 10h!

* Processo de Instalação:

1 - Iniciar o Rasp com a imagem de instalação no cartão SD

2 - Configurar a hora do sistema (“date MMDDhhmmYYYY”)

3 - Configurar as partições do “disco”, opcional (“cfdisk /dev/mmcblk0”)

4 - Se a fonte dos pacotes for um pendrive:

Montar o pendrive

#mkdir /source

#mount /dev/sda1 /source

5 - Iniciar a ferramenta de instalação

#setup

Obs.: Quando questionado, indicar a 1ª partição “/dev/mmcblk0p1”, FAT32, criada pela imagem de boot, como “/boot”.

Obs2: NÃO REINICIAR!!!!!

6 – Remover kernel :

```
root@Raspi:# ROOT=/mnt removepkg \
kernel_kirkwood kernel-modules-kirkwood \
kernel_tegra kernel-modules-tegra \
kernel_versatile kernel-modules-versatile
```

7 – Instalar kernel para o rasp, se encontra dentro do diretório “raspi-extra” da fonte de instalação :

```
root@Raspi:# mount -t vfat /dev/mmcblk0p1 /mnt/boot
root@Raspi:# ROOT=/mnt installpkg /raspi-extra/kernel* /raspi-extra/raspi*
```

Obs3: pode reiniciar!

* Pós-Instalação

Atualizando o Firmware

(foi necessário reinstalar “ca-certificates”)

```
root@Raspi:# wget http://www.cbpf.br/~juliana/files/rpi-update
root@Raspi:# chmod +x rpi-update
root@Raspi:# rpi-update
```

Após atualizar o Firmware, disponível o comando “vcgencmd”***

```
root@raspberrypi:~# /opt/vc/bin/vcgencmd measure_temp
temp=48.2’C
```

root@raspberrypi:~# /opt/vc/bin/vcgencmd measure_volts core
volt=1.21V

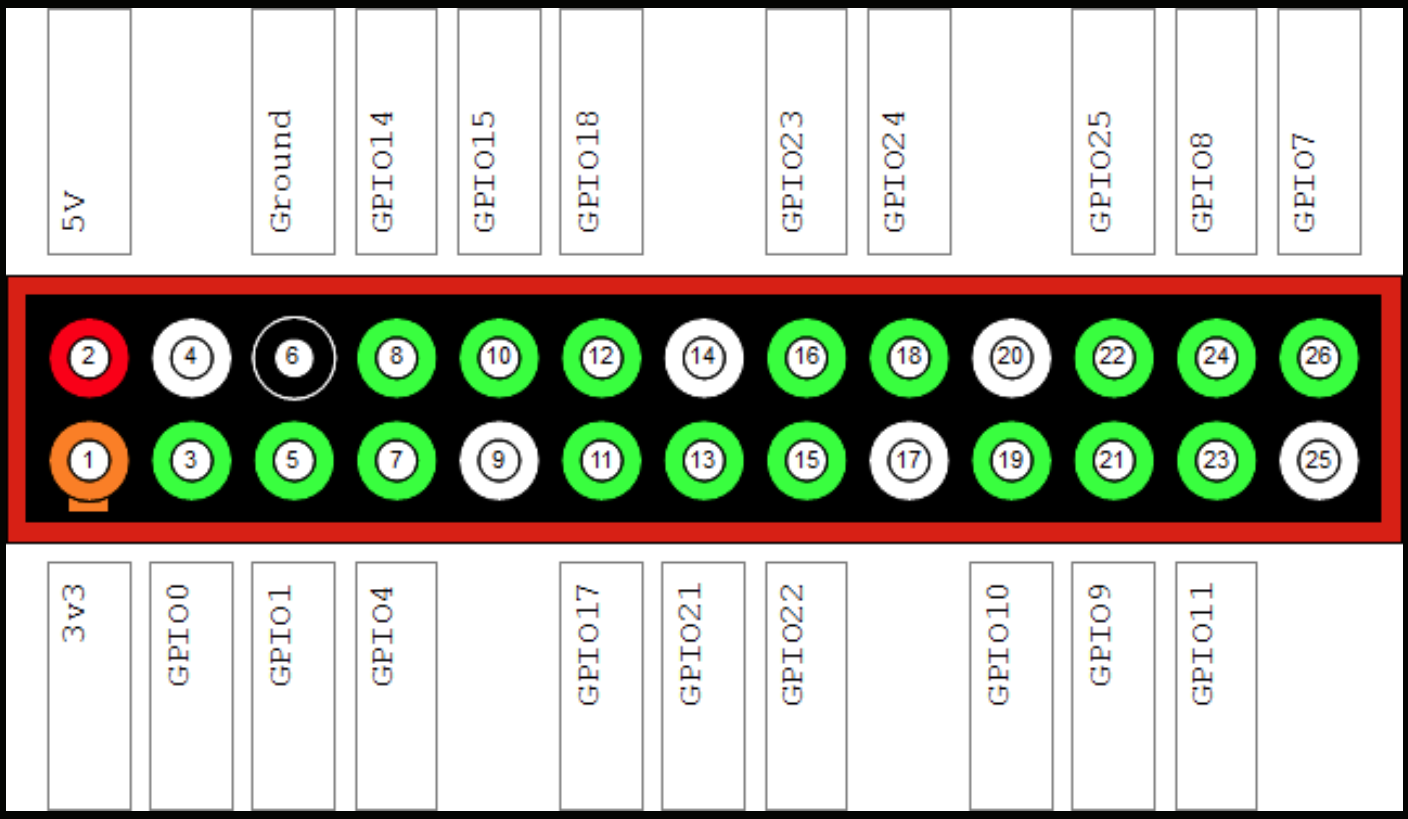
root@raspberrypi:~# /opt/vc/bin/vcgencmd measure_clock arm
frequency(45)=700074000

root@raspberrypi:~# /opt/vc/bin/vcgencmd codec_enabled MPG4
MPG4=enabled

***http://elinux.org/RPI_vcgencmd_usage

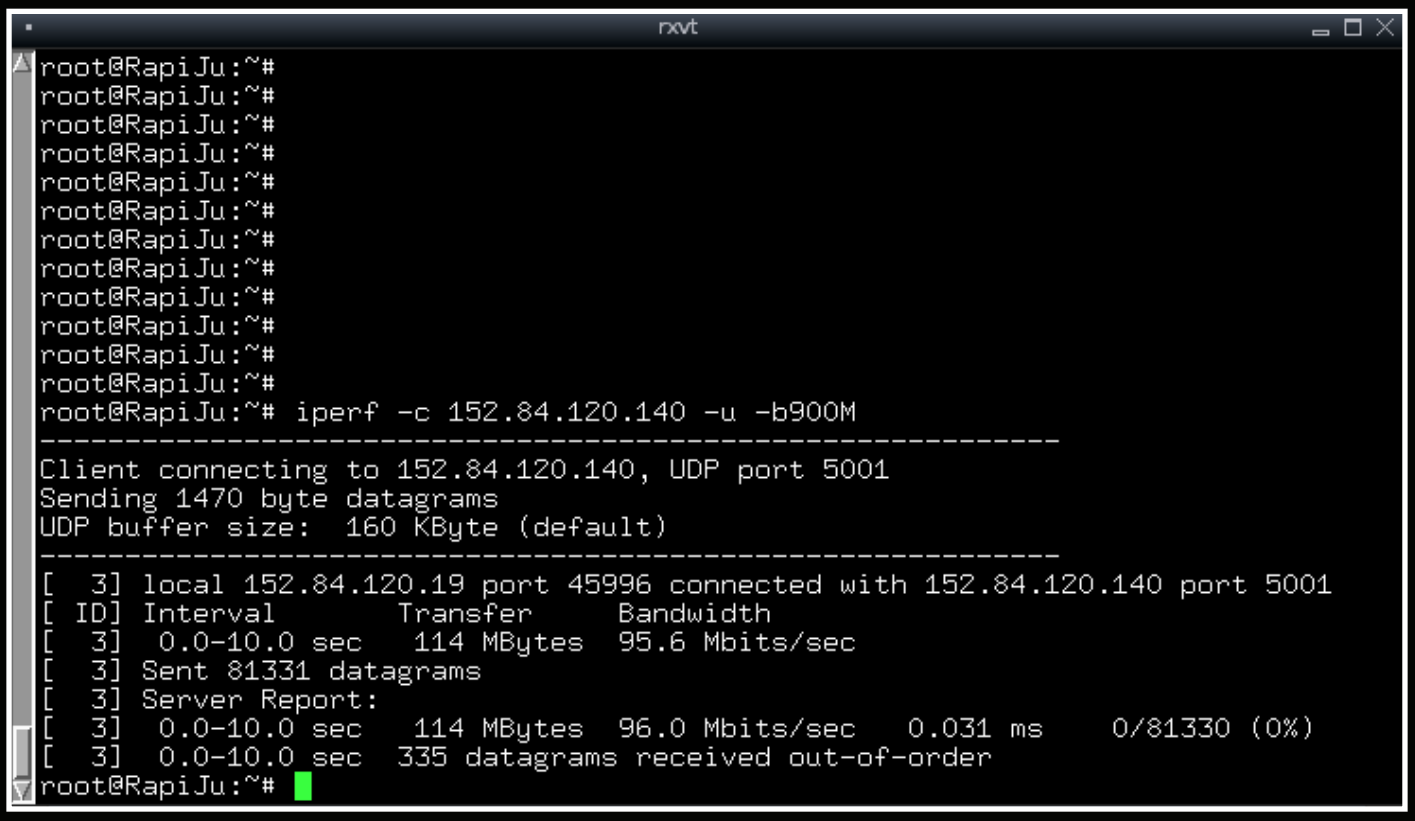
- GPIO

* Acesso e controle da interface GPIO



- Resultados

* Hardware



```
                  L M B E N C H 3 . 0 S U M M A R Y
                 ------------------------------------
                 (Alpha software, do not distribute)

  Basic integer operations - times in nanoseconds - smaller is better
 ----------------------------------------------------------------------
                Host OS intgr intgr intgr intgr intgr
                          bit add mul div mod
      --------- ------------- ------ ------ ------ ------ ------
          RapiJu Linux 3.2.27- 1.4800 1.5400 7.4200 111.8 49.7
   Basic float operations - times in nanoseconds - smaller is better
  ----------------------------------------------------------------
                      Host OS float float float float
                                add mul div bogo
           --------- ------------- ------ ------ ------ ------
               RapiJu Linux 3.2.27- 45.6 39.7 179.8 406.1

                                CPU (lmbench):
                              Int - 675,67 Mips
                             Float - 21,93 Mflops
                           CPU - bogomips: 697,95

Benchmark completo em - http://www.cbpf.br/~juliana/docs/raspi-imbench.txt
```

* GPIO	
C (gcc + bcm2835.h) - 3.800Hz	Bash - 1635Hz


```
root@Raspi# while true ; do echo 1 > value ; echo 0 > value ; done
```

- Conclusão

* Monitoração:

  Necessidade de se inserir um hardware em locais de difícil acesso

  Com seu tamanho reduzido, é fácil perceber que um hardware como o RaspiberryPi pode executar tarefas computacionais em locais impossíveis de se alocar um PC comum.

* Atuação:

  Facilidade promovida pelo time de desenvolvimento do Kernel Linux no acesso e controle das interfaces GPIO - baixo custo de desenvolvimento (diminuto período de desenvolvimento).

* União de “controle de sinais elétricos” + “acesso remoto via rede” (web)

  Integração do interfaceamento elétrico externo com as ferramentas de controle remoto via rede já existente (ssh, vnc, etc).

* Possibilidade de alimentação via cabo de rede (PoE)

  Alimentado por fonte única de 5Vcc.

* Possibilidade de alimentação por painéis solares / baterias

  Baixíssimo consumo (2,5w ~ 3,5w)

* Atrativos adicionais:

  - Dispensa dissipação mecânica ativa

  - Leve

  - Baixo custo

  - Nível zero de ruído

  - Resistente a extremas forças G

- Referências

* arm.slackware.com
* elinux.org/RPi_config.txt
* arm.slackware.com/introduction
* elinux.org/RPI_vcgencmd_usage
* iperf.fr
* www.bitmover.com/lmbench
* www.broadcom.com/products/BCM2835
* www.kernel.org/doc/Documentation/gpio.txt
* www.intel.co.jp/content/dam/www/public/us/en/documents/application-notes/chipset-gpio-sw-config-appl-note.pdf

- Agradecimentos

Patrick Volkerding;
Linus Torvalds;
Richard Stallman;
Jon Maddog Hall;
Pank;
Raphael Bastos;
Renato Gravino;
David Coverdale;
Jesé Henrique Campos “Canisso”;
Rodrigo Aguiar “Digão”;
Chester Bennington;
Paul McCoy;
Klaus Meine;
Juliana Silvestre;
Joaquim (da Silva Chavier);
Irmãos Cavalera;
dentre outros que contribuem para um mundo melhor.

Slackware à Prova de Bala

Slackware (Atualmente Kernel 3.2.29 x86_64) se destaca dos demais sistemas operacionais por ser criado pensando em administradores que dominam o seu sistema, neste mundo não procuramos por Kernel carregado de módulos e drivers que nunca iremos usar e, ou pacotes prontos para aplicação que queremos usar. No mundo Slackware recebemos um Kernel limpo com poucos módulos / drivers carregados em memória, nós mesmos criamos nossos pacotes e compilamos ao nosso modo, gerando assim, um ambiente familiar ao administrador.

Partindo deste ponto, irei levantar aspectos básicos sobre como melhorarmos ainda mais a segurança do nosso Slackware evitando que script kiddies, crackers e possíveis ameaças que possam comprometer seu funcionamento.

0x00 - Partições

Crie partições separadas e aplique políticas especificas a cada uma delas (rw, suid, dev, exec, auto, nouser).

```
/home
/tmp
/usr
/var
```

Ex:
Edite o arquivo "`/etc/fstab`" com seguintes configurações:

```
/home defaults,nodev
/tmp defaults,nodev,nosuid,
 noexec
/usr defaults,nodev
/var defaults,nodev
```

0x01 - Remova Pacotes não utilizados

Para isso você poderá utilizar a ferramenta:

```
#pkgtool
```

0x02 - Desative serviços não utilizados

Serviços carregados via inetd no diretório / arquivo (/etc/inetd.conf)

Scripts do sistema carregados no diretório (/etc/rc.d/)

```
#chmod -x rc.nomedoscript
```

0x03 - Controle de acesso

Para prevenir acesso não autorizado ao seu computador:

Hosts.allow - Arquivo aonde contém os serviços que certo IPs podem acessar(`/etc/hosts.allow`)

Hosts.deny - Arquivo que contém os hosts que NÃO podem acessar a determinados serviços(`/etc/hosts.deny`)

0x04 - Senhas Fortes

Sempre utilize senhas fortes (`/etc/passwd`) e valide suas senhas utilizando o John The Ripper.

https://github.com/magnumripper/JohnTheRipper

0x05 - Sudo

Utilize o “sudo” para que apenas o “root” possa fazer “su” no teu sistema e “sudo” para usuários e ferramentas que precisam de root, esta configuração devera ser feita em “`/etc/sudoers`”.

0x06 - Acesso Remoto (SSH)

Edite o arquivo abaixo para realizar as configurações de acesso via ssh.

`/etc/ssh/sshd_config`

Parâmetros:

- Port 22 * Você ainda pode alterar a porta padrão para outra Ex: 55
- Protocol 2
- PermitRootLogin no * Proibindo acesso dó root via SSH
- AllowUsers Logan * Qual Usuário poderá acessar via SSH

Para serviços como http podemos ainda ocultar banner e também criar configurações personalizadas que irão melhorar sua segurança.

Exemplos de como ocultar banners em:

apache

http://www.sysadminworld.com/2012/hide-the-server-version/

nginx

http://nginxlibrary.com/hide-nginx-version/

0x07 - Subindo um Firewall iptables simples e objetivo

Use o iptables para criação de regras de firewall e NATs.

#mkdir /etc/firewall

#vim slack_fw.sh

```
---Inicio da Config
#!/bin/sh
# Slackware Firewall

PATH='/sbin'
INET='eth1'
LOCAL='172.16.1.0/24'
LOOP='127.0.0.0/8'
HOST='172.16.1.2/32'

echo “Iniciando Firewall ...”

echo “1” > /proc/sys/net/ipv4/conf/all/rp_filter

/usr/sbin/iptables -F
/usr/sbin/iptables -X
/usr/sbin/iptables -Z

/usr/sbin/iptables -P FORWARD DROP
```

```
/usr/sbin/iptables -P INPUT
DROP
/usr/sbin/iptables -A INPUT -p
icmp -j DROP


/usr/sbin/iptables -A
INPUT -m state --state
ESTABLISHED,RELATED -j ACCEPT
/usr/sbin/iptables -A INPUT -s
$LOOP -d $LOOP -i lo -j ACCEPT
/usr/sbin/iptables -A INPUT
-s $HOST -p tcp --dport 55 -j
ACCEPT
/usr/sbin/iptables -A INPUT -s
$LOCAL -p icmp --icmp-type 8 -m
state --state NEW -j DROP
/usr/sbin/iptables -A INPUT -s
$LOCAL -p icmp --icmp-type 0 -m
state --state NEW -j DROP


echo " Firewall Started !!!"
---Fim da Config
```

Crie um script para carregar o teu firewall:

cd /etc/rc.d

#vi rc.firewall

--- Inicio da Config

```
# Script de Carga do Firewall

firewall_start() {
  sh /etc/firewall/slack_fw.sh
}

firewall_stop() {
  iptables -t nat -F
  iptables -F
}
```

```
firewall_restart() {
  firewall_stop
  sleep 2
  firewall_start
}

case "$1" in
'start')
  firewall_start
  ;;
'stop')
  firewall_stop
  ;;
'restart')
  firewall_restart
  ;;
*)
firewall_start
esac
--- Fim da Config


#chmod +x rc.firewall
#vi rc.local


---Inicio da Config
/etc/rc.d/rc.firewall start
--- Fim da Config
```

Agora Inicie o seu Firewall:

Ex:

```
#cd /etc/rc.d/
#./rc.firewall start
```

0x08 - Atualização

Utilize sempre a versão current e aplique os patches !!!

0x09 - Ferramentas prontas

Script para Hardening criado pelo Thiago Laurito:

http://slackzine.com.br/hardening_slack.php

GRSecurity Path para Hardening do Kernel:

http://grsecurity.net

0x10 - Testando o ambiente

As ferramentas abaixo são boas opções automatizadas para teste do Sistema. Ficando ainda a cargo do administrador realizar testes com ferramentas próprias e ou técnicas mais especificas a este ambiente.

Chkrootkit - Ferramenta que vai te ajudar no processo de detecção de rootkits.

ftp://ftp.pangeia.com.br/pub/seg/pac/chkrootkit.tar.gz

RKHunter - Outra opção para detecção de rootkits.

http://sourceforge.net/projects/rkhunter/files/

Aide - Verifica alterações (permissões UID,GID) no Sistema.

http://aide.sourceforge.net

OSSEC - HIDS que poderá te ajudar verificando vulnerabilidade em seu servidor

http://www.ossec.net/

0x11- Conclusão

Aconselho fortemente uma boa leitura na documentação das ferramentas listadas neste artigo antes de por em produção.

Referencias:

http://www.linuxquestions.org

http://slackware.com

http://slackdummies.blogspot.com.br

Author(s) HICKS, A.; LUMENS C.; CANTRELL, D.; JOHNSON, L. - Book The Official Guide to Slackware Linux 2ª ed.Brentwood, CA, 2005

Autor: Ricardo Logan

Pós-graduado em Segurança da Informação, especialista em redes Lan/Wan, implementando soluções de rede e segurança. Slackuser desde 2006 e entusiasta em pesquisas com malware, pentest e forense.

Nginx com vatapá e camarão

O Nginx é um servidor web leve e versátil muito utilizado como proxy reverso suportando diversos protocolos como HTTP/HTTPS, imap, pop3 e SMTP. Neste artigo apresentarei as diversas formas de como configurar e blindar (hardening) de forma simples e efetiva.

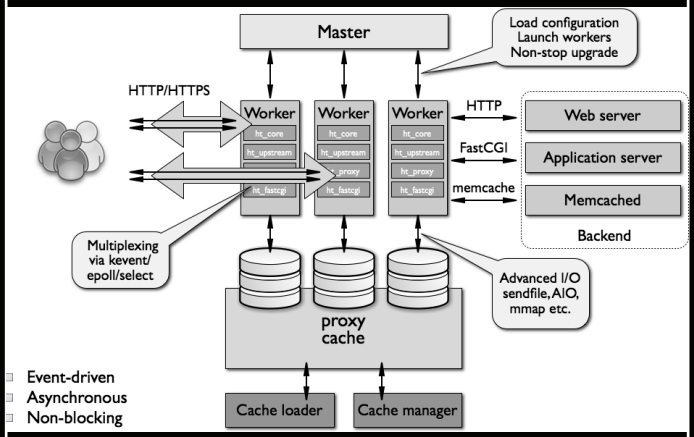
- Introdução

Criado por Igor Sysoev o nginx segue em crescente adoção, algumas características que o difere de seus concorrentes são:

- Arquitetura orientada a alta perfor mance;
- Pouco consumo de memória;
- Proxy reverso com caching.

Segundo estatísticas da Netcraft, ele é o servidor web usado em 41% dos 12 milhões de web sites hospedados na Amazon. O suporte a load balance com in-band health check permite a criação de uma infraestrutura com total tolerância a falhas e alta disponibilidade, sua arquitetura segue um modelo assíncrono, modular usando notificação de eventos, single-thread com multiplexação dedicando tarefas específicas para diferentes processos otimizando a utilização de CPU e memória.

Figura 1: Nginx architecture



- Proxy Reverso

O Nginx é reconhecido mundialmente como a melhor solução open source de proxy reverso existente no mercado, pelo suporte a diversos sistemas operacionais, alta performance e facilidade de configuração. Nesse tópico apresentarei a instalação e configuração de um proxy reverso incluindo o hardening necessário.

Em todos os testes foi utilizado a versão 14.0 do Slackware e a versão estável do nginx (1.4.2) disponível no momento da criação deste artigo.

- Instalação

O processo de instalação é bastante simples necessitando somente instalar as dependências, baixar o pacote nginx-1.4.2-i486-1sl.txz disponível em http://slack.isper.sk/pub/slackware-14.0/net-

work/ e instalá-lo usando a ferramenta installpkg.

Instale as seguintes dependências usando o slackpkg:

- zlib
- libxml2-2.8.0-i486-2_slack14.0
- gd-2.0.35-i486-4gd-2.0.35-i486-4
- libXpm-3.5.10-i486-1
- libX11-1.5.0-i486-1
- fontconfig-2.9.0-i486-1
- libxcb-1.8.1-i486-1
- libXau-1.0.7-i486-1
- libXdmcp-1.1.1-i486-1
- aspell-0.60.6-i486-1
- aspell-en-6.0_0-noarch-4
- enchant-1.5.0-i486-1
- icu4c-49.1.2-i486-1
- libmcrypt-2.5.8-i486-1
- libxslt-1.1.26-i486-2
- t1lib-5.1.2-i486-3

```
alexos@wotan:~# wget http://slackware.org.uk/slacky/slackware-14.0/network/GeoIP/1.4.8/GeoIP-1.4.8-i486-2sl.txz
alexos@wotan:~# wget http://slack.isper.sk/pub/slackware-14.0/network/nginx/1.4.2/nginx-1.4.2-i486-1sl.txz
alexos@wotan:~# sudo installpkg nginx-1.4.2-i486-1sl.txz GeoIP-1.4.8-i486-2sl.txz
```

- Configuração

Toda as configurações tanto do serviço quanto dos vhosts ficam centralizadas no arquivo /etc/nginx/nginx.conf .

```
alexos@wotan:~# cd /etc/nginx
alexos@wotan:~# sudo mv nginx.conf nginx.conf.orig
alexos@wotan:~# sudo vim nginx.conf
user nobody;

#Total de threads. Configure de acordo com a quantidade de CPU existente, acima de 2 CPUs = 4.
worker_processes  4;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

#Juntamente com o work_processes permite calcular o máx. de clientes ( max clients = worker_processes * worker_connections )
events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
```

```
    access_log  /var/log/nginx/
access.log  main;

    sendfile        on;

    tcp_nodelay on;

    gzip  on;
    gzip_disable “MSIE [1-6]\.
(?!.*SV1)”;

  include /etc/nginx/conf.d/*.conf;

# Protecao contra DoS
   client_body_buffer_size 1K;
   client_header_buffer_size 1k;
   client_max_body_size 2M;
   large_client_header_buffers 2
1k;
   client_body_timeout 10;
   client_header_timeout 10;
   keepalive_timeout 5 5;
   send_timeout 10;

# Oculta banner
  server_tokens off;

# Limita o maximo de conexoes concor-
rentes por IP
limit_conn_zone $binary_remote_
addr zone=addr:10m;
limit_conn addr 10;

# Logar ip do backend
    proxy_set_header   Host
$host;
    proxy_set_header   X-Real-IP
$remote_addr;
    proxy_set_header   X-For-
warded-For  $proxy_add_x_for-
warded_for;
   proxy_max_temp_file_size 0;
    proxy_connect_timeout
90;
    proxy_send_timeout
90;
    proxy_read_timeout
90;
    proxy_buffer_size
4k;
    proxy_buffers                4
32k;
    proxy_busy_buffers_size
64k;
    proxy_temp_file_write_size
64k;
    proxy_cache_methods GET HEAD
POST;

# === vhost ACME ===

server {
    listen       80;
    server_name  acme.local;

    access_log  /var/log/nginx/
acme.access.log  main;
    error_log  /var/log/nginx/
acme.error.log;

    location / {
     proxy_pass
http://192.168.0.2/;
     proxy_next_upstream error
timeout invalid_header http_500
http_502 http_503 http_504;
     proxy_redirect off;
     proxy_buffering off;
     proxy_set_header
Host            $host;
     proxy_set_header
```

```
X-Real-IP       $remote_addr;
    proxy_set_header
X-Forwarded-For $proxy_add_x_
forwarded_for;
        }
   }
}
```

Após salvar o arquivo verifique se toda a configuração está correta.

```
alexos@wotan:~# sudo nginx -t
nginx: the configuration file /
etc/nginx/nginx.conf syntax is
ok
nginx: configuration file /etc/ng-
inx/nginx.conf test is success-
ful
```

Agora basta iniciar o serviço e seu proxy reverso está pronto para ser utilizado.

```
alexos@wotan:~# sudo chmod +x /
etc/rc.d/rc.nginx
alexos@wotan:~# sudo /etc/rc.d/
rc.nginx start
```

- Alta disponibilidade

O Nginx também pode ser utilizado como um sistema capaz de prover alta disponibilidade e balanceamento de carga. Habilitar esse recurso é muito simples bastando para isso incluir a diretiva upstream no vhost do site, como mostra o exemplo abaixo:

```
upstream acme {
        hash_ip;
        server 192.168.0.2;
        server 192.168.0.3;
        server 192.168.0.4;
        }
```

- Prioridade e Failover

Para definir a prioridade de cada back-end utiliza-se o parâmetro weight o valor padrão é 1. No exemplo abaixo as 3 primeiras requisições serão enviadas para o servidor 192.168.0.2, a 4a e 5a para o 192.168.0.3 e a 6a. para o 192.168.0.4.

```
upstream acme {
        hash_ip;
        server 192.168.0.2
weight=3;
        server 192.168.0.3
weight=2;
        server 192.168.0.4;
        }
```

O failover é habilitando usando os parâmetros max_fails define o total de falhas na requisição e fail_timeout define o intervalo de tempo entre as falhas a partir dai a requisição é enviada para o próximo backend.

```
upstream acme {
        hash_ip;
        server 192.168.0.2
max_fails=3  fail_timeout=30s;
        server 192.168.0.3;
        server 192.168.0.4;
        server 192.168.0.5
down;
```

```
        }
```

O parâmetro down é utilizado quando um backend está inacessível.

Exemplo de vhost usando failover

```
server {
    listen       80;
    server_name  acme;

upstream acme {
        hash_ip;
        server 192.168.0.2
max_fails=3  fail_timeout=30s;
        server 192.168.0.3;
        server 192.168.0.4;
        server 192.168.0.5
down;
        }

    access_log  /var/log/ng-
inx/acme.access.log  main;
    error_log  /var/log/nginx/
acme.error.log;

    location / {
     proxy_pass  http://acme/;
     proxy_next_upstream er-
ror timeout invalid_head-
er http_500 http_502 http_503
http_504;
     proxy_redirect off;
     proxy_buffering off;
     proxy_set_header
Host            $host;
     proxy_set_header
X-Real-IP       $remote_addr;
     proxy_set_header
X-Forwarded-For $proxy_add_x_
forwarded_for;
        }
```

- Webserver com suporte PHP

Outra forma de utilização do Nginx é como um simples webserver hospedando aplicações escritas em diversas linguagens como PHP ou Ruby. Neste tópico descrevo como configurar o suporte ao PHP utilizando o fastcgi.

Como todos os softwares necessários foram instalados basta apenas configurar o vhost e testar.

Exemplo de um vhost hospedando uma aplicação PHP

```
server {
    listen       80;
    server_name  acme;
    root /var/www/;
    index  index.php;

    access_log  /var/log/ng-
inx/acme.access.log  main;
    error_log  /var/log/nginx/
acme.error.log;

    location / {
    try_files $uri $uri/ /in-
dex.php$is_args$args;
     }

#Habilitando o suporte ao PHP
    location ~ \.php$ {
        fastcgi_pass
127.0.0.1:9000;
        fastcgi_param
SCRIPT_FILENAME $document_
root$fastcgi_script_name;
        include        fastc-
gi_params;
    }
```

```
# == Restrições de acesso ==


#htaccess
 location ~ /\.ht {
        deny  all;
   }

#Ambiente administrativo
location /admin {
root /var/www/acme/;
index index.php;
allow 200.222.222.222;
deny all;
}

#Arquivos
location =/*.txt {
deny all;
log_not_found off;
access_log off;
}

location =/xmlrpc.php{
deny all;
log_not_found off;
access_log off;
}

location =/readme.html {
deny all;
log_not_found off;
access_log off;
}


   }


}
```

Teste o arquivo de configuração e reinicie os serviços

```
alexos@wotan:~# sudo nginx -t
alexos@wotan:~# sudo chmod +x
/etc/rc.d/rc.php-fpm
alexos@wotan:~# sudo /etc/
rc.d/rc.nginx restart && sudo
/etc/rc.d/rc.php-fpm restart
```

Para validar crie um arquivo index.php dentro do diretório /var/www com o código abaixo e tente acessar usando o navegador.

```
<html>
<body>
<h1>Wotan!</h1>
<?php
print “<h2>Hello World!</h2>”
?>
</body></html>
```

- Conclusão

O artigo descreve como é simples configurar o Nginx para atender diversas necessidades, durante sua utilização a robustez e otimização no uso dos recursos são facilmente perceptíveis. Recomendo a integração com um WAF (e.g. Modsecurity ou Naxsi) proporcionando uma camada extra de proteção para as aplicações web.

Alexandro Silva
< alexos@alexos.org>

- Referências

http://nginx.org/

http://wiki.nginx.org/Main

http://www.aosabook.org/en/nginx.html

Configure uma replicação simples de arquivos via Csync2 no Slackware

- Descrição

- Sincronização de arquivos no servidor com csync2.
- Ele possui vários algoritmos de sincronia, como pelo arquivo mais novo, mais antigo, maior, entre outros. Nesse caso estarei usando o mais novo (younger)
- Além de sincronizar é possível executar comandos remotos, como reiniciar o apache após um rotate de log.
- Pode ser utilizado com N servidores.

- Softwares e Versões

- Csync2 1.3X
- Slackware Linux 13.37

Instale as dependências

- librsync
- sqlite2
- sqlite3
- gnutls
- flex
- bison

Instale o libtasn1

```
$ wget ftp://ftp.gnutls.org/pub/gnutls/libtasn1/libtasn1-1.4.tar.gz
$ ./configure; make ; make install
```

- Configuração

Instalação

Obtenha o código fonte do Csync2:

http://oss.linbit.com/csync2/

Compilar o Csync2:

```
$ ./configure
$ make
$ make install
$ make cert
```

Criar o /etc/csync2.cfg:

```
#Exemplo /etc/csync2.cfg
group mygroup
{
        host web2xt-master@10.199.199.1 web2xt-slave@10.199.199.2;

        key /etc/csync2.key_hackstore;

        include /etc/passwd;
        include /etc/group;
        include /etc/shadow;

        auto younger;
```

```
}
```

Criar a chave de segurança em algum dos nodos:

```
$ csync2 -k /etc/csync2.key_hackstore
```

Copiar a chave de segurança e ssl para os demais hosts:

```
$ scp /etc/csync2* root@outroshosts:/etc
```

Caso deseje criar certificado na mao:

```
$ openssl genrsa -out /etc/csync2_ssl_key.pem 1024


$ openssl req -new -key /etc/csync2_ssl_key.pem -out /etc/csync2_ssl_cert.csr


$ openssl x509 -req -days 600 -in /etc/csync2_ssl_cert.csr -signkey /etc/csync2_ssl_key.pem -out /etc/csync2_ssl_cert.pem
```

Setar permissões de conf:

```
$ chmod 640 /etc/csync2.cfg
```

Copiar arquivo de conf e chave para o outro servidor.

Arquivo de init

Crie o arquivo de init (slackware):

```
vi /etc/rc.d/rc.csync2

#!/bin/sh

#

# “$Id: csync2.sh.in 6649 2013 coffnix $”

#

#   Startup/shutdown script for Csync2 - slackzine

#

#   Maiores infos: www.hackstore.com.br - www.slackzine.com.br

#

#### OS-Dependent Information

#

#   Linux chkconfig stuff:

#

#   chkconfig: 235 99 00

#   description: Startup/shutdown script for Csync2

#


#

#   NetBSD 1.5+ rcorder script lines.  The format of the following two

#   lines is very strict -- please don’t add additional spaces!

#

# PROVIDE: csync2

# REQUIRE: DAEMON

#


#### OS-Dependent Configuration


case “`uname`” in

        Linux*)

                IS_ON=/bin/true

                if test -f /etc/init.d/functions; then

                        . /etc/init.d/functions


ECHO=echo

ECHO_OK=”echo_success”
```

```
ECHO_ERROR="echo_failure"
                else

ECHO=echo

ECHO_OK=:

     ECHO_ERROR=:

                fi

                ;;

        *)

                IS_
ON=/bin/true

                ECHO=echo

                ECHO_OK=:

                ECHO_ERROR=:

                ;;

esac

#### OS-Independent Stuff

for file in /etc/TIME-
ZONE /etc/rc.config /
etc/sysconfig/clock; do

        if test
-f $file; then

                . $file

        fi
done

if test "x$ZONE" != x; then

        TZ="$ZONE"

fi

if test "x$TIME-
ZONE" != x; then

        TZ="$TIMEZONE"

fi
if test "x$TZ" != x; then
        export TZ
fi
#
# Don't use TMPDIR envi-
ronment variable from init
script, as that can
# cause csync2d to set Temp-
Dir to a user's temporary
directory instead
# of the default...
#

unset TMPDIR

#
# Make sure we have the
standard program directories
in the path
# since some operating sys-
tems (this means YOU HP-UX!)
don't
# provide a standard path on
boot-up...
#
```

```
if test "x$PATH" = x; then
        PATH="/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/sbin:/usr/local/bin"
else
        PATH="/usr/local/sbin:/usr/local/bin:/bin:/usr/bin:/sbin:/usr/sbin:$PATH"
fi

export PATH
#
# See if the CSYNC2 server (csync2d) is running...
#

case "`uname`" in
        Linux* | *BSD* | Darwin*)
                pid=`ps ax | awk '{if (match($5, ".*/csync2d$") || $5 == "csync2d") print $1}'`
                ;;
        *)
                pid=""
                ;;
esac
#
# Start or stop the CSYNC2 server based upon the first argument to the script.
#
case $1 in
        start | restart | reload)
                if $IS_ON csync2; then
                        if test "$pid" != ""; then

kill -HUP $pid
                        else

prefix=/usr/local

exec_prefix=/usr/local

/usr/local/sbin/csync2

if test $? != 0; then

$ECHO_FAIL

$ECHO "csync2: unable to $1 scheduler."

exit 1

fi
                        fi

$ECHO_OK

$ECHO "csync2: ${1}ed scheduler."
                fi
                ;;

        stop)
                if test "$pid" != ""; then
                        killall -TERM /usr/local/sbin/csync2

$ECHO_OK

$ECHO "csync2: stopped scheduler."
```

```
                fi
                ;;
        status)
                if test
"$pid" != ""; then
                        echo
"csync2: scheduler is run-
ning."
                else
                        echo
"csync2: scheduler is not
running."
                fi
                ;;

        *)
                echo "Us-
age: csync2 {reload|re-
start|start|status|stop}"
                exit 1
                ;;
esac
#
# Exit with no errors.
#
exit 0
#
# End of "$Id: csync2.sh.in
6649 2013 21:46:42Z mike $".
#
```

Setar permissões e colocar no boot:

```
chmod 755 /etc/rc.d/
rc.csync2
echo "/etc/rc.d/rc.csync2
start" >> /etc/rc.d/rc.lo-
cal
```

Iniciar o daemon em ambas as máquinas:

```
/etc/rc.d/rc.csync2 start
```

Sincronizar:

```
$ csync2 -x
```

Na primeira sincronia pode gerar erros. Rodar mais de uma vez nesse caso.

Colocar o script no crontab com a frequência de sincronismo:

```
#Crontab
#Sync files
*/30 * * * *   root  /usr/
local/sbin/csync2 -x
```

- Biografia

Raphael Bastos, conhecido como Ch3z, Chemonz ou Coffnix.

* Estudante de Engenharia Eletrônica e de Telecomunicação, fundador do Grupo de Usuários Slackware de Minas Gerais (GUS-MG), membro do Grupo de Usuários Slackware do Brasil (GUS-BR), realizador dos eventos II Oficina Livre, Slackware Show Brasil e BHACK Conference, Desenvolvedor Funtoo Linux, fundador do Área 31 Hackerspace, criador e mantenedor do Yaxkin Linux (Gentoo for i686 and x86_64), criador e mantenedor do Kankin Linux (Funtoo for ARMv6 and ARMv7), editor e mantenedor da SlackZine, mantenedor do site hackstore.com.br. trabalha atualmente com consultoria de servidores, Linux, Unix, *BSD, bancos de dados, altíssima disponibilidade, capacity planning, replicação, integração de ambientes e/ou sistemas operacionais, segurança da informação, MTA, redes, virtualização, cloud computing e clusters.

Enumeração de portas HTTP, análise de vulnerabilidade e screenshot com NMAP

By: Gr1nch DcLabs Security Team

Um dos primeiros passos de um pentest interno é enumerar os ativos de rede e quais portas estão abertas nestes ativos. Portanto neste pequeno artigo, vamos focar apenas na enumeração e análise de vulnerabilidades de dispositivos que estão executando o protocolo HTTP com a ferramenta NMAP, levando em consideração que você já deve ter ouvido falar dessa ferramenta algumas vezes.

Caso não ouviu falar recomendo a leitura do manual do NMAP para melhor compreensão deste artigo.

Bom, alguns administradores infelizmente ainda acreditam que trocar a porta padrão do serviço é o suficiente para proteger o serviço de um ataque. Portanto vamos fazer um scan completo em todos os dispositivos da rede em todas as portas do protocolo TCP e filtrar apenas os IP's que estão com portas HTTP abertas.

Já fiz isso várias vezes e percebi que em alguns casos, quando você pede o NMAP para verificar todas as portas (p165535) de um range inteiro, ele pode não verificar todas as portas. Outros amigos também comentaram que frequentemente enfrentam o mesmo problema.

Segue um exemplo de sintax do NMAP no qual mesmo especificando todas as portas e todo o range de IP's, o NMAP pode não fazer a análise completa:

```
#nmap –A –sT –p165535
172.28.10.0/24 –oA relatorio_
nmap –statsevery10s
```

Eu particularmente prefiro dividir essa tarefa do NMAP em algumas etapas garantindo que todo o range de portas e IP's serão verificados. Sendo as etapas:

- Levantamento dos hosts ativos na rede.
- Levantamento de quais portas HTTP estão ativas nestes hosts.
- Análise de vulnerabilidade individual de cada porta e screenshot do website.

Vou descrever separadamente cada etapa de um script simples que utilizo para execução sistemática de cada uma destas etapas.

- 1º Etapa) Levantamento de hosts ativos na rede:

Suponto que estamos avaliando a rede 172.28.10.0/24 e todos seus ativos de rede; vamos executar

a seguinte sintax do NMAP para levantarmos apenas as máquinas atualmente ativas na rede:

```
#nmap T5 sP 172.28.10.0/24 |
awk '{print $5}' | grep 172 |
sed 's/(//g' | sed 's/)//g' |
sed '/^$/d' | tee hosts_ati-
vos.txt
```

Basicamente o comando acima verifica quais são as máquinas ativas na rede filtrando apenas a coluna que possui os IP's e separando este resultado em um arquivo chamado (hosts_ativos.txt).

Observação: A utilização do comando "awk" neste caso é para separarmos a coluna que contém os IP's, esse resultado pode mudar de acordo com a versão do NMAP que você esta executando e também de acordo com as flags que você ativou para o NMAP, portanto fique atento.

- 2ª Etapa) Levantamento de quais portas HTTP estão ativas nestes hosts:

O modo que proponho para garantir que a análise será realizada em todas as portas dos IP's ativos em nossa rede, é executar o NMAP individualmente para cada IP que detectamos em nossa rede no passo anterior. Para isso vamos utilizar o "while" que percorre cada linha do arquivo (hosts_ativos.txt) criado no passo anterior e executa o NMAP em cada um dos IP's contidos neste arquivo, lembrando que o objetivo é listar apenas as portas abertas que estão executando o protocolo HTTP. Segue a sintax:

```
while read line
do
echo "=-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=-=-=-=-=-=-=-=-=-=-
=-=-="
echo "Procurando portas com
serviço HTTP/HTTPS no IP:
$line "
nmap -sT -sV -p1 -65535 $line
| grep -iE 'nmap scan re-
port|http |httpd ' |tee -a
hosts_com_http.txt
done < hosts_ativos.txt
```

No exemplo acima, estou utilizando a opção -sT para verificar em quais portas podemos nos conectar via TCP e tambem vamos verificar a versão do serviço que esta sendo executado na porta, portanto também precisamos utilizar a opção sV. Como o objetivo é varrer todas as portas do protocolo TCP, pois não sabemos em qual delas o administrador configurou o serviço, vamos definir o range de portas completo do TCP com a opção p165535.

Como o objetivo é listar apenas as portas que estão executando o protocolo HTTP e qual é máquina que esta executando este protocolo vamos utilizar uma expressão regular simples para filtrar o resultado com o grep. Para isso utilizei a sintax:

```
grep -iE 'nmap scan re-
port|http |httpd '
```

Finalmente o script separa essa relação de IP's e portas com serviço HTTP no arquivo:

```
hosts_com_http.txt
```

Agora precisamos fazer um parse simples no arquivo (`hosts_com_http.txt`) para separarmos apenas a lista de portas que vamos procurar vulnerabilidades. No script eu armazeno isso temporariamente na variável PORTAS_HTTP separando cada porta por vírgula para utilizarmos essa variável na sintax do NMAP:

```
PORTAS_HTTP=`cat hosts_com_
http.txt | awk '{print $1}'|
cut -d / -f 1 | grep -iv
"nmap" | sort | uniq |sed
':s;N;s/\n/,/;bs'`
```

- 3ª Etapa) Análise de vulnerabilidade individual de cada porta e screenshot do website:

O NMAP possui uma série de scripts auxiliares que o transformam em uma espécie de canivete suiço.

Em 14 de junho de 2012, Ryan Linn membro do time de pentesters da SpiderLabs desenvolveu um script para o NMAP que permite que você tire um screenshot do website que está em execução no IP que você esta análisando.

Antes de utilizarmos este script precisamos baixar alguns pré requisitos conforme Ryan descreve em seu post:

```
http://blog.spiderlabs.
com/2012/06/usingnmaptoscreen-
shotwebservices.
html
```

Segue a sequência de comandos para download e instalação do script:

```
#cd /tmp
#wget
http://wkhtmltopdf.googlecode.
com/files/wkhtmltoimage0.11.0_
rc1-statici386.tar.bz2
#tar -jxvf wkhtmltoim-
age-0.11.0_
rc1-statici-386.tar.bz2
#cp wkhtmltoimage-i386 /usr/
local/bin/
#git clone git://github.com/
SpiderLabs/NmapTools.git
#cd NmapTools/NSE/
#cp http-screenshot.nse /usr/
local/share/nmap/scripts/
#nmap --script-updatedb
```

Obs: Os comandos acima foram realizados em uma máquina 32Bits para máquinas 64bits utilize este link:

```
https://code.google.com/p/
wkhtmltopdf/downloads/de-
tail?name=wkhtmltoim-
age-0.11.0_rc-1static-amd64.
tar.bz2
```

você também deverá fazer ajustes no script. Siga as instruções deste link para maiores informações:

```
http://www.pentestgeek.
com/2012/07/11/usingn-
map-to-screenshotweb-ser-
vices-troubleshooting/
```

Após termos instalado o script para o screenshot e atualizado a base de dados do NMAP, vamos executar

a análise de vulnerabilidade, acessar e tirar screenshot de todas as portas que estão com o protocolo HTTP habilitado com a sintax:

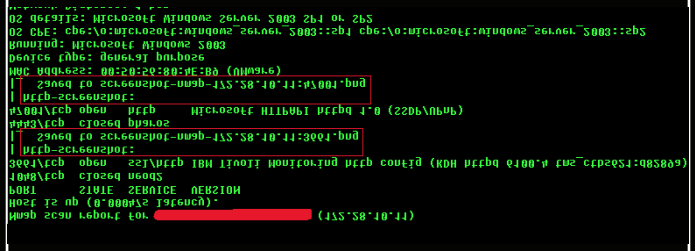
```
while read line
do
echo "=-=-=-=-=-=-=-=-=-=-=-=-
=-=-=-=-=-=-=-=-=-=-=-=-=-
=-=-="
echo "Varrendo portas do IP:
$line "
nmap -A -p$ PORTAS_HTTP
--script=version,vuln,http-
screenshot $line -oA relato-
rio_portas_http --stats-every
10s
done < hosts_ativos.txt
```

Novamente utilizei o WHILE para percorrer todas as linhas do arquivo hosts_ativos.txt, porém dessa vez verificando

versão, vulnerabilidades e tirando o screenshot das portas (`--script=version,vuln,httpscreenshot`).

O script também gera o relatório final do NMAP (-oA relatorio_portas_http) que você poderá utilizar em outros aplicativos como o MetaSploit por exemplo.

O script do NMAP httpscreenshot gera um arquivo com a extenção PNG para cada porta com protocolo HTTP que ele acessou e tirou screenshot:



Você pode gerar um HTML para visualizar de maneira mais fácil os screenshots gerados:

```
printf "<HTML><BODY><BR>" >
preview.html

ls -1 *.png | awk -F : '{
print $1":"$2"\n<BR><IMG
SRC=\""$1"%3A"$2"\"

width=400><BR><BR>"}' >> pre-
view.html

printf "</BODY></HTML>" >>
preview.html
```

Após gerar o HTML basta utilizar um navegador qualquer para analisar os resultados:

Segue agora o script completo:

```
#!/bin/bash

nmap -T5 -sP 172.28.10.0/24 |
awk '{print $5}' | grep 172 |
sed 's/(//g' | sed 's/)//g' |
sed '/^$/d' | tee hosts_ati-
vos.txt

while read line

do

   echo "=-=-=-=-=-=-=-=-=-=-=-
   =-=-=-=-=-=-=-=-=-=-=-=-=-
   =-=-=-="

   echo "Procurando portas com
   serviço HTTP/HTTPS no IP:
   $line"

   nmap -sT -sV p165535 $line
   | grep -iE 'nmap scan re-
   port|http |httpd ' |

tee -a hosts_com_http.txt

done < hosts_ativos.txt

PORTAS_HTTP=`cat hosts_com_
```

```
http.txt | awk '{print $1}'|
cut -d
/ -f 1 | grep -iv "nmap" |
sort | uniq |sed ':s;N;s/\
n/,/;bs'`
while read line
do
  echo "=-=-=-=-=-=-=-=-=-=-
  =-=-=-=-=-=-=-=-=-=-=-=-=-
  =-=-=-=-="
  echo "Varrendo portas do IP:
  $line "
  nmap -A -p$PORTAS_HTTP
  --script=version,vuln,http-
  screenshot $line -oA
relatorio_portas_http
--stats-every 10s
done < hosts_ativos.txt
printf "<HTML><BODY><BR>" >
preview.html
ls -1 *.png | awk -F : '{
print $1":"$2"\n<BR><IMG
SRC=\""$1"%3A"$2"\"
width=400><BR><BR>"}' >> pre-
view.html
printf "</BODY></HTML>" >>
preview.html
```

- Biografia

Rêner Alberto (aka Gr1nch) é membro do grupo de pentesters DcLabs Security Team e também membro fundador do Área31 Hackerspace, atua como pentester e analista de segurança da informação em uma instituição financeira do Brasil. E já palestrou em alguns eventos de segurança no Brasil como BHack, ValeSecurity Conference, Websecurity Forum e Bsides São Paulo.

Contatos: gr1nch@dclabs.com.br

irc.freenode.net /j #dclabs, #area31