Mikrotik port knocking extention - labean

После публикации на тему port knocking один читатель поделился шикарной утилитой, с помощью которой можно похожим образом открывать доступ на основе HTTP запроса, что во многих случаях удобнее отправки пакетов. Программа называется labean (https://github.com/uprt/labean или см. ниже). Автор русскоязычный, поэтому подобное название не случайно (кто не понял, читайте наоборот).

Идея подобного функционала у меня давно сидела в голове, но до реализации дело не дошло. А готовых инструментов я раньше не встречал и даже не слышал о них. Labean работает очень просто и эффективно. С его помощью можно выполнить любое действие при определённом http запросе. Например, обращаемся на url http://10.20.1.56/labean/tuktuk/ssh/on, а labean выполняет проброс порта на нужный сервер. То есть выполняет конкретное действие:

```
iptables -t nat -A PREROUTING -p tcp --dport 31004 -i ens18 \
-s 10.20.1.1 -j DNAT --to 10.30.51.4:22
```

Я проверил работу labean, очень понравился результат. Настраивается быстро и легко. Рассказываю по шагам.

Сначала собираем утилиту из исходников. Написана на GO.

```
# apt install golang git nginx
# git clone https://github.com/uprt/labean.git
# cd labean
# go build
```

Копируем бинарник, конфиг и systemd unit.

```
# cp labean /usr/sbin
# cp examples/labean.conf.ex /etc/labean.conf
# cp examples/labean.service.ex /etc/systemd/system/labean.service
```

Рисуем примерно такой конфиг:

```
{
 "listen": "127.0.0.1:8080",
 "url_prefix": "tuktuk",
 "external_ip": "10.20.1.56",
 "real_ip_header": "X-Real-IP",
 "allow_explicit_ips": true,
 "tasks": [
  {
   "name": "ssh",
   "timeout": 30,
   "on_command": "iptables -t nat -A PREROUTING -p tcp --dport 31004 -i ens18 -s {clientIP} -j
DNAT --to 10.30.51.4:22",
   "off_command": "iptables -t nat -D PREROUTING -p tcp --dport 31004 -i ens18 -s {clientIP} -j
DNAT --to 10.30.51.4:22"
  },
  {
   "name": "postfix",
   "timeout": 0,
   "on_command": "systemctl start postfix",
   "off_command": "systemctl stop postfix"
  }
  ]
}
```

Не буду все настройки описывать, можно в репозитории посмотреть. Я для теста сделал 2 сервиса и 2 разных действия для них. Первое действие добавляет правило в iptables, второе - запускает службу. Проверил оба примера, всё работает. Привожу для наглядности, чтобы вы понимали функционал. Labean может делать всё, что угодно, а не только правила firewall изменять.

Теперь для любого хоста, к которому мы будем обращаться, добавляем в конфиг nginx ещё один location. Я добавил прямо в default, чтобы по ip обращаться.

```
location ~ ^/labean/(.*) {
  proxy_set_header X-Real-IP $remote_addr;
  proxy_pass http://127.0.0.1:8080/$1;
}
```

При желании можно добавить basic auth и закрыть паролем.

Перечитываем службы systemd и запускаем:

```
# systemctl daemon-reload
# systemctl start nginx labean
```

Идём по урлу `http://10.20.1.56/labean/tuktuk/ssh/on` и смотрим результат:

```
{
 "commandLine": "iptables -t nat -A PREROUTING -p tcp --dport 31004 -i ens18 -s 10.20.1.1 -j DNAT
--to 10.30.51.4:22",
 "returnCode": 0,
 "timeoutInSeconds": 30,
 "clientIp": "10.20.1.1"
}
```

Зайдём в консоль сервера и посмотрим правила iptables:

```
# iptables -L -v -n -t nat | grep 31004
  0   0 DNAT    tcp -- ens18 *    10.20.1.1      0.0.0.0/0      tcp dpt:31004 to:10.30.51.4:22
```

Через 30 секунд правило исчезнет. За это отвечает параметр timeout. Если поставить 0, то само удаляться не будет, и нужно будет вручную зайти на закрывающий урл - http://10.20.1.56/labean/tuktuk/ssh/off.

Как вы уже поняли, обращение к url `http://10.20.1.56/labean/tuktuk/postfix/on` и `http://10.20.1.56/labean/tuktuk/postfix/off` будет запускать и останавливать службу postfix. Дергать эти адреса можно как в браузере, так и через curl.

В документации заявлена ещё одна возможность через дополнительный аргумент ?ip=123.56.78.9 указывать для обработки различные ip адреса, но у меня почему-то сходу не заработало. Пока не разбирался.

Вот такая простая и удобная программа. Рекомендую обратить внимание. Я точно буду пользоваться, так как это очень удобно. Заметка получилось полной инструкцией.

#security #gateway

LABEAN

https://github.com/uprt/labean#readme

What is port knocking?

Port knocking is a method of externally opening ports on a firewall by doing some actions, e.g. generating a connection attempt on a set of prespecified closed ports. [Wikipedia]

The primary purpose of port knocking is to prevent an attacker from scanning a system for potentially exploitable services by doing a port scan, because unless the attacker sends the correct knock sequence, the protected ports will appear closed. Another purpose is to disguise some services (e.g. VPN or proxy) running on the server. For example, the server receives connections on 443 port and acts as a usual HTTPS web server, but after knocking it will modify firewall rules to allow specified IP to connect to another service on the same port - for third-party observers (like corporate or ISP Deep-Packet-Inspection systems) it will look exactly like ordinary HTTPS.

Why Labean?

Classic implementations of port knockers (like knockd) allow a client to open port or start services by generating a connection attempt on a set of prespecified closed ports. This is simple and usually reliable, but there are some tricky cases. First of all, this requires using special clients or scripts on the client device (including mobile gadgets or network routers). More significant problem is that all ports and protocols except standard 80 (HTTP) and 443 (HTTPS) can be banned on corporate or ISP firewall, so you can't use 'classic' knocking in this case. So that's why Labean was created.

How it works?

Briefly: there is a front-end web server (like nginx or lighttpd) running on your VDS/VPS/etc. and it serves some ordinary web content like cute kittens' videos, Linux distros' ISOs or Wikipedia mirror. But when you want to connect to the hidden service (VPN, proxy, ssh daemon, etc.), you perform GET request (using CURL or a web browser) like:

```
https://someserver.org/secret/vpn/on,
```

process basic authentication, then your front-end web server reverse-proxies the request to Labean, and Labean starts service or applies firewall rules to open access exclusively for your IP address. When you finished working with hidden service, you just make GET

```
https://someserver.org/secret/vpn/off
```

and the access becomes closed.

Another way is to use *automatic timeout control*.

For example, if you only need to open firewall port temporarily to accept a connection, you can set a timeout for 30 or 60 seconds. After making GET request you start your client and initiate a connection to the hidden service, then the timeout expires and the rule is deleted. Your established connection is still active, but no one else even from your IP can establish a new one without knocking again.

Building and installation

Labean is written in Go language, so you'll need Golang compiler installed in your system. Please refer to official Go website and your distro manpages to get and install Golang compiler. I tried to build Labean with Go 1.6 and 1.10 and everything was fine, perhaps it will work with even older versions.

```
git clone https://github.com/uprt/labean.git
cd labean
go build
```

If everything is okay, you'll have `labean` binary executable in your current directory.

I recommend putting `labean` binary to `/usr/sbin` directory and `labean.conf` to `/etc/` (see the next chapter about this configuration file).

Labean can be started simply:

```
./labean <path_to_config_file>
```

After starting `labean` will send its logs to syslog service, so you can check them out in /var/log/syslog or /var/log/messages file depending on your distro.

But the better way is to run it as a service. If your distro uses LSB-compatible init system or Systemd, feel free to use sample service files that you can find in `examples` dir in the source tree:

```
cp ./examples/labean.init.ex /etc/init.d/labean
/etc/init.d/labean start
```

```
update-rc.d labean defaults
# or...
cp ./examples/labean.service.ex /etc/systemd/system/labean.service # this path can be different
in your distro
systemctl daemon-reload
systemctl start labean
systemctl enable labean
```

Another important step of Labean installation is a configuration of your frontend web server (reverse-proxy). Labean does not support SSL, HTTP auth and serving static or dynamic websites itself. I like "UNIX-way" philosophy: "Make each program do one thing well." So we have to use a web server with Labean. It should perform the following things:

- serve HTTPS (SSL) connections with valid certificates
- require basic HTTP authorization for 'secret' URL
- add 'X-Real-IP" or similar header to an HTTP request
- pass the request to Labean (running on another TCP port like 8080) for 'secret' URL

I recommend using Nginx for this. The simplest variant of the config you can find it in `examples` directory of the source tree. Anyway, you can use any other webservers/reverse-proxies if you want.

Config file

```
Labean config file is a simple JSON document.
{
  "listen": "127.0.0.1:8080",
  "url_prefix": "secret",
  "external_ip": "192.30.253.113",
  "real_ip_header": "X-Real-IP",
  "allow_explicit_ips": false,
  "tasks": [
    {
      "name": "vpn",
      "timeout": 30,
      "on_command": "iptables -t nat -A PREROUTING -p tcp -s {clientIP} --dport 443 -j REDIRECT -
-to-port 4443",
      "off_command": "iptables -t nat -D PREROUTING -p tcp -s {clientIP} --dport 443 -j REDIRECT
--to-port 4443"
    },
    {
      "name": "sshd",
      "timeout": 0,
      "on_command": "/etc/init.d/sshd start",
      "off_command": "/etc/init.d/sshd stop"
    }
    ]
}
```

Here is the description of its fields:

- `"listen":` IP address and port to listen for incoming connections from reverse proxy;
- `"url_prefix"`: if your reverse-proxy can't rewrite URLs, you can set the prefix to trim;
- `"external_ip"`: IP of your server. This is not 'must-have' option, its just a sugar to replace {serverIP} in the commands strings if you need;
- `"real_ip_header"`: the name of HTTP header with the real client IP added by the reverse-proxy (usually it is "X-Real-IP");
- `"allow_explicit_ips"`: if true, you can explicitly specify client IP address in GET request, like `https://someserver.org/secret/service/off/?ip=123.56.78.9`. This can be helpful when you established VPN connection and later want to manually de-activate secret service using Labean's internal (local) IP inside the tunnel. This feature allows you to stop services started by other users, so use it carefully, and it is disabled by default.
- `"tasks"`: the array of the 'tasks' to start or stop hidden services;

- "name": the unique name of the hidden service. You will use it in your HTTP GET queries: `https://someserver.org/secret/<name>/{on|off}`;
- "timeout": timeout to automatically switch your hidden service or firewall rule "off" after "on". If it is set to 0, it means that timeout feature will be disabled and you need to "off" your service manually;
- "on_command": command line to start your service or activate firewall rule. You can use `{serverIP}` and `{clientIP}` macro in it, they will be automatically replaced by the corresponding values;
- "off_comand": the same as "on_command", but does exactly the opposite thing :)