Tunnel TCP connections through ICMP.

BSD-3-Clause license

446 stars 74 forks 9 watching Branches Activity
Tags

Public repository

6 Branches 10 Tags Go to file t Go to file Add file Code

EMREOYUN --list-libpcap-devices fixed (#35) b1baa74 · 11 months ago

contrib	added systemd conf/service file	7 years ago
debian	debian/rules: package systemd files	6 years ago
selinux	updated selinux policy file and added...	6 years ago
src	--list-libpcap-devices fixed (#35)	11 months ago
test	ignore incomplete packets instead of...	5 years ago
web	removed CVS leftover..	8 years ago
.clang-format	initial clang-format file	5 years ago
.dockerignore	docker	5 years ago
.gitlab-ci.yml	fixed archlinux ci build	5 years ago
AUTHORS	updated AUTHORS (masaq-, elnerd)	6 years ago
COPYING	copyright update	7 years ago
ChangeLog	1.42-release	6 years ago
Dockerfile	use tini init	5 years ago
Makefile.am	ptunnel-ng:	8 years ago
NEWS	ptunnel-ng:	8 years ago
PKGBUILD	ArchLinux PKGBUILD md5 chksm upd...	6 years ago
PKGBUILD.dev	added PKGBUILD dev version (builds ...	6 years ago
README	README's provide some simple ptun...	5 years ago
README.md	Removed Travis CI.	2 years ago
autogen.sh	autogen.sh can now be executed fro...	5 years ago
	configure.ac: enable ASAN, LSAN and...	6 years ago

configure.ac

model_file.c	added coverity model file	7 years ago

README | BSD-3-Clause license

PingTunnel-[N]ew[G]eneration Read Me

What is ptunnel-ng?

```
Ptunnel-NG is a bugfixed and refactored version of Ptunnel with some additional
features e.g. change the magic value without recompiling.
```

What is ptunnel?

```
Ptunnel is an application that allows you to reliably tunnel TCP connections
to a remote host using ICMP echo request and reply packets, commonly known as
ping requests and replies.
```

Simple usage

Opens a SSH over ICMP tunnel to a remote.

Server:

```
sudo ptunnel-ng
```

Client:

```
sudo ptunnel-ng -p[Server-IP/NAME] -l2222
```

```
ssh -p2222 -luser 127.0.0.1
```

Restricted usage

Opens a SSH over ICMP tunnel to a remote but restricts destination IP/Port for tunnel clients. 10.0.3.1 is the machine your SSH daemon listens on. This can be a virtual machine, container or (.*).

Server:

```
sudo ptunnel-ng -r10.0.3.1 -R22
```

Client:

```
sudo ptunnel-ng -p[Server-IP/NAME] -l2222 -r10.0.3.1 -R22
```

```
ssh -p2222 -luser 127.0.0.1
```

Reverse shell usage

Opens a SSH over ICMP tunnel to a remote and creates a reverse tunnel on same but restricts destination IP/Port for tunnel clients. 10.0.3.1 is the machine your SSH daemon listens on. This can be a virtual machine, container or (.*).

Server:

```
sudo ptunnel-ng -r10.0.3.1 -R22
```

Client:

```
sudo ./src/ptunnel-ng -p[Server-IP/NAME] -l2222 -r10.0.3.1 -R22
```

```
ssh -R 127.0.0.1:22222 127.0.0.1 -p2222
```

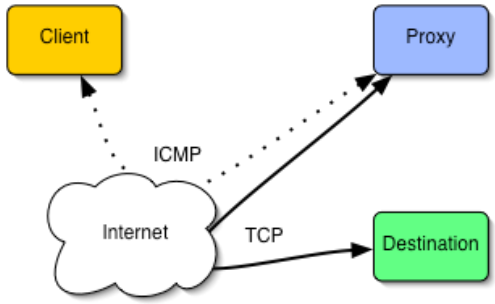
Server:

```
ssh -p22222 -luser 127.0.0.1
```

How does it work?

```
ICMP Packet structure
```


```
Ptunnel program setup
```



Contact details

```
The ptunnel-ng fork was done by Toni Uhlig:
   <matzeton@googlemail.com>
You can contact the author of ptunnel, Daniel Stoedle, here:
   <daniels@cs.uit.no>
The official ptunnel website is located here:
   <http://www.cs.uit.no/~daniels/PingTunnel/>
```

Dependencies

```
Required: pthread
Optional: pcap, selinux
```

Compiling

```
Either run "./autogen.sh" for a fully automatic build or run it manually with:
    "./configure && make"

You should end up with a binary called ptunnel-ng.
This serves as both the client and proxy. You can
optionally install it using "make install".
To compile the Windows binary. You will need mingw installed.
If you want pcap support you will need the WinPcap library as well.
WinPcap is available here:
  <http://www.winpcap.org/install/bin/WpdPack_4_0_2.zip>

REMEMBER: ptunnel-ng might not work on Windows without WinPcap!
```

Running

```
Ptunnel works best when starting as root, and usually requires starting as root.
Common ptunnel-ng options:

Proxy(Server):
        ./ptunnel-ng -r<destination address> -R<destination port> -v <loglevel>
                     -P<password> -u<user> -g<group>

Forwarder(Client):
        ./ptunnel-ng -p <address> -l <listen port> -r<destination address>
                     -R<destination port> -v <loglevel>
                     -P<password> -u<user> -g<group>

The -p switch sets the address of the host on which the proxy is running. A
quick test to see if the proxy will work is simply to try pinging this host -
if you get replies, you should be able to make the tunnel work.
If pinging works but you are not able to establish a tunnel, you should play
around with the -m switch and change the magic value. A IDS/IPS or Firwall
might try to fool you.

The -l, -r and -R switches set the local listening port, destination address
and destination port. For instance, to tunnel ssh connections from the client
machine via a proxy running on proxy.pingtunnel.com to the computer
login.domain.com, the following command line would be used:

sudo ./ptunnel-ng -p proxy.pingtunnel.com -l 8000 -r login.domain.com -R 22

An ssh connection to login.domain.com can now be established as follows:

ssh -p 8000 localhost

If ssh complains about potential man-in-the-middle attacks, simply remove the
offending key from the known_hosts file. The warning/error is expected if you
have previously ssh'd to your local computer (i.e., ssh localhost), or you have
used ptunnel-ng to forward ssh connections to different hosts.

Of course, for all of this to work, you need to start the proxy on your
proxy-computer (we'll call it proxy.pingtunnel.com here). Doing this is very
simple:

sudo ./ptunnel-ng

If you find that the proxy isn't working, you will need to enable packet
capturing on the main network device. Currently this device is assumed to be
an ethernet-device (i.e., ethernet or wireless). Packet capturing is enabled by
```

```
giving the -L switch, and supplying the device name to capture packets on (for
instance eth0 or en1). The same goes for the client. On versions of Mac OS X
prior to 10.4 (Tiger), packet capturing must always be enabled (both for proxy
and client), as resent packets won't be received otherwise.

To protect yourself from others using your proxy, you can protect access to it
with a password using the -P switch. The password is never sent in
the clear, but keep in mind that it may be visible from tools like top or ps,
which can display the command line used to start an application.

Finally, the -u switch will attempt to run the proxy in unprivileged mode (i.e.,
no need for root access), and the -v switch controls the amount of output from
ptunnel-ng. -1 indicates no output, 0 shows errors only, 1 shows info messages, 2
gives more output, 3 provides even more output, level 4 displays debug info and
level 5 displays absolutely everything, including the nasty details of sends and
receives. The -o switch allows output to be saved to a logfile.

Security features: Please see the ptunnel-ng man-page for instructions.
```

Supported operating systems

```
Ptunnel supports most operating systems with libpcap, the usual POSIX functions
and a BSD sockets compatible API. In particular, it has been tested on Linux
Fedora Core 2 and Mac OS X 10.3.6 and above. As of version 0.7, ptunnel-ng can also
be compiled on Windows, courtesy of Mike Miller, assuming mingw and WinPcap is
installed.
```

TODOs

```
- refactoring
- libsodium integration
```

Credits and contributors

```
Daniel Stoedle et al.
```

License

```
Ping Tunnel NG is Copyright (c) 2017-2019, Toni Uhlig <matzeton@googlemail.com>,
```

Releases 9

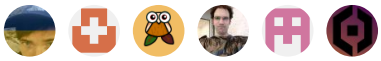
Last Release (for a long time) Latest
on Nov 27, 2024

+ 8 releases

Packages

No packages published

Contributors 6



Languages

C 79.7% HTML 9.5% M4 3.8% Roff 3.5% Shell 2.5% Makefile 0.9% Dockerfile 0.1%