pike Module

Bogdan Iancu
FhG FOKUS

Edited by
Bogdan Iancu

pike Module

Edited by and Bogdan Iancuand Bogdan Iancu

Revision History

Revision $Revision: 1.1 $ $Date: 2003/07/16 17:25:09 $

Table of Contents

List of Examples

Chapter 1. User's Guide

1.1. Overview

The module keeps trace of all (or selected ones) incoming requests's IP source and blocks the ones that exceeded some limit. Works simultaneous for IPv4 and IPv6 addresses.

1.2. Dependencies

1.2.1. SER Modules

The following modules must be loaded before this module:

- *No dependencies on other SER modules*.

1.2.2. External Libraries or Applications

The following libraries or applications must be installed before running SER with this module loaded:

- *None*.

1.3. Exported Parameters

1.3.1. `sampling_time_unit` (integer)

Time period used for sampling (or the sampling accuracy ;-)). The smaller the better, but slower. If you want to detect peeks, use a small one. To limit the access (like total number of requests on a long period of time) to a proxy resource (a gateway for ex), use a bigger value of this parameter.

Default value is 2.

Example 1-1. Set `sampling_time_unit` parameter

```
...
modparam("pike", "sampling_time_unit", 10)
...
```

1.3.2. `reqs_density_per_unit` (integer)

How many requests should be allowed per sampling_time_unit before blocking all the incoming request from that IP. Practically, the blocking limit is between (let's have x=reqs_density_per_unit) x and 3*x for IPv4 addresses and between x and 8*x for ipv6 addresses.

Default value is 30.

Example 1-2. Set `reqs_density_per_unit` parameter

```
...
modparam("pike", "reqs_density_per_unit", 30)
...
```

1.3.3. `remove_latency` (integer)

For how long the IP address will be kept in memory after the last request from that IP address. It's a sord of timeout value.

Default value is 120.

Example 1-3. Set `remove_latency` parameter

```
...
modparam("pike", "remove_latency", 130)
...
```

1.4. Exported Functions

1.4.1. `pike_check_req()`

Process the source IP of the current request and returns false if the IP was exceeded the blocking limit.

Meaning of the parameters is as follows:

Example 1-4. `pike_check_req` usage

```
...
if (!pike_check_req()) { break; };
...
```

Chapter 2. Developer’s Guide

One single tree (for both IPv4 and IPv6) is used. Each node contains a byte, the IP addresses stretching from root to the leafs.

Example 2-1. Tree of IP addresses

```
   / 193 - 175 - 132 - 164
tree root /                  \ 142
  \ 195 - 37 - 78 - 163
   \ 79 - 134
```

To detect the whole address, step by step, from the root to the leafs, the nodes corespoding to each byte of the ip address are expanded. In order to be expended a node has to be hit for a given number of times (possible by different addresses; in the previous example, the node “37” was expended by the 195.37.78.163 and 195.37.79.134 hits).

For 193.175.132.164 with x= reqs_density_per_unit:

- After first req hits -> the “193” node is built.
- After x more hits, the “175” node is build; the hits of “193” node are splited between itself and its child--both of them gone have x/2.
- And so on for node “132” and “164”.
- Once “164” build the entire address can be found in the tree. “164” becomes a leaf. After it will be hit as a leaf for x times, it will become “RED” (further request from this address will be blocked).

So, to build and block this address were needed 3*x hits. Now, if reqs start comming from 193.175.132.142, the first 3 bytes are alredy in the tree (they are shared with the previous address), so I will need only x hits (to build node “142” and to make it “RED”) to make this address also to be blocked. This is the reason for the variable number of hits necesary to block an IP.

The maximum number of hits to turn an address red are (n is the address’s number of bytes):

1 (first byte) + x (second byte) + (x / 2) * (n - 2) (for the rest of the bytes) + (n - 1) (to turn the node to red).

So, for IPv4 (n = 4) will be 3x and for IPv6 (n = 16) will be 9x. The minimum number of hits to turn an address red is x.

Chapter 3. Frequently Asked Questions

1. Where can I find more about SER?

Take a look at http://iptel.org/ser.

2. Where can I post a question about this module?

First at all check if your question was already answered on one of our mailing lists:

- http://mail.iptel.org/mailman/listinfo/serusers
- http://mail.iptel.org/mailman/listinfo/serdev

E-mails regarding any stable version should be sent to <`serusers@iptel.org`> and e-mail regarding development versions or CVS snapshots should be send to <`serdev@iptel.org`>.

If you want to keep the mail private, send it to <`serhelp@iptel.org`>.

3. How can I report a bug?

Please follow the guidelines provided at: http://iptel.org/ser/bugs