

Add support for IPv6

master

v1.0.1

...

v1.0.0.0

Browse files

parsonsbots

committed 6 days ago

1 parent [fb58870](#)

commit [01e8f123ed85817e54960304a6fa4a90bcf9fb69](#)

Showing 4 changed files with 233 additions and 134 deletions.

Unified

Split

173 README.md

@@ -1,7 +1,7 @@

1 # Dedicated IP Checker

2

3 ## Metadata

4 - * Version 1.0.0

5 * Author: Will Parsons

6 * Website: https://parsonsbots.com

7

13

14 ## Changelog:

15 * 1.0.0: Initial release for IPv4 only

16

17

18 ## What is a Dedicated IP Checker?

19 This simple dedicated IP checker was built to be integrated with hosting and VPN / proxy services to provide a way for users to verify dedicated IP address exclusivity.

45 ## Usage

46 ### Define IP List to Check

47 /**

48 - * IPv4 list to check separated by new line

49 - * This list should be submitted through a public HTML form textarea field

50 *

51 - * Reserved IPv4 addresses below are used for demo purposes

52 *

53 */

54 - \$ipv4ListToCheck = implode("\n", array(

55 '172.16.88.10',

56 '172.16.8.3',

57 '172.16.200.3',

67 '10.5.30.105',

68 '10.5.30.106',

69 '10.5.31.108',

70 - '_10.5.30.199:80;', // Invalid

71 IP formats will be parsed

72 '10.5.89.44',

73 '10.8.8.100',

74 '10.8.8.101',

1 # Dedicated IP Checker

2

3 ## Metadata

4 + * Version 1.0.1

5 * Author: Will Parsons

6 * Website: https://parsonsbots.com

7

13

14 ## Changelog:

15 * 1.0.0: Initial release for IPv4 only

16 + * 1.0.1: Add support for IPv6

17

18 ## What is a Dedicated IP Checker?

19 This simple dedicated IP checker was built to be integrated with hosting and VPN / proxy services to provide a way for users to verify dedicated IP address exclusivity.

46 ## Usage

47 ### Define IP List to Check

48 /**

49 + * IP list to check separated by new line

50 *

51 + * Reserved IP addresses below are used for demo purposes

52 *

53 */

54 + \$ipListToCheck = implode("\n", array(

55 + '2001:db8:abcd:0008:847g:3e2:1088:8888',

56 + '2001:db8:abcd:0008::ffff:ffff',

57 + '2001:db8:abcd:0008:0000:0000:8888:ffff',

58 + '2001:db8:abcd:0008::1234:8888:FFFF',

59 '172.16.88.10',

60 '172.16.8.3',

61 '172.16.200.3',

71 '10.5.30.105',

72 '10.5.30.106',

73 '10.5.31.108',

74 + '_10.5.30.199;', // Invalid IP

75 formats will be parsed

76 '10.5.89.44',

77 '10.8.8.100',

78 '10.8.8.101',

```

77
78 ### Define IPs That Belong to Users
79 /**
80 - * List of dedicated IPv4 addresses that belong to
each user (IPv4 => UUID format)
81 * Users should have access to see their own user
ID
82 *
83 - * Reserved IPv4 addresses below are used for demo
purposes
84 *
85 */
86 - $userIps = array(
87 - '192.168.0.2' => '5c95396e-97c8-49af-abb3-
17e04221ccee',
88 - '192.168.0.3' => '5c95396e-97c8-49af-abb3-
17e04221ccee',
89 - '10.5.31.108' => '5c95354f-43c8-4976-b69d-
12334221ccee',
90 - '172.16.88.10' => '5c953514-e9e8-4cca-
954c-12334221ccee',
91 - '172.16.200.3' => '5c953514-e9e8-4cca-
954c-12334221ccee',
92 - '10.5.89.44' => '5c952170-f908-45e9-af43-
6bf64221ccee',
93 - '10.5.30.199' => '5c9518c1-0ad8-4e41-80a1-
5bd54221ccee',
94 - '172.16.8.3' => '5c951454-0418-4ca5-80f8-
5b6b4221ccee',
95 - '192.168.40.4' => '5c951242-56a9-401a-
8d74-5b7b4221ccee',
96 - '10.8.8.102' => '5c95396e-97c8-49af-abb3-
17e04221ccee'
97 - );
98
99 - // Include formatted sample data (this file should
be changed to your own database values and user input)
100 - require_once('data.php');
101
102 - // Include dedicated IP checker
103 - require_once('checker.php');
104
105 - $checker = new DedicatedIpChecker();
106 - $verifiedIpv4List = $checker-
>verify($ipv4ListToCheck, $userIps);
107
108 - ### Output
109 - echo '<pre>';
110
111 - foreach ($verifiedIpv4List as $ip => $userId) {
112 - echo 'IP: ' . $ip . "\n";
113 - echo 'User ID: ' . $userId . "\n\n";
114 - }
115
116 - echo '</pre>';
117
118 - /* Output:

```

```

81
82 ### Define IPs That Belong to Users
83 /**
84 + * List of dedicated IP addresses that belong to
each user (IP => UUID format)
85 * Users should have access to see their own user
ID
86 *
87 + * Reserved IP addresses below are used for demo
purposes
88 *
89 */
90 + $userIps = array(
91 + '2001:db8:abcd:0008:0000:0000:8888:ffff' => '5c9518c1-
0ad8-4e41-80a1-5bd54221ccee',
92 + '192.168.0.2' => '5c95396e-97c8-
49af-abb3-17e04221ccee',
93 + '192.168.0.3' => '5c95396e-97c8-
49af-abb3-17e04221ccee',
94 + '10.5.31.108' => '5c95354f-43c8-
4976-b69d-12334221ccee',
95 + '172.16.88.10' => '5c953514-e9e8-
4cca-954c-12334221ccee',
96 + '172.16.200.3' => '5c953514-e9e8-
4cca-954c-12334221ccee',
97 + '10.5.89.44' => '5c952170-f908-
45e9-af43-6bf64221ccee',
98 + '10.5.30.199' => '5c9518c1-0ad8-
4e41-80a1-5bd54221ccee',
99 + '172.16.8.3' => '5c951454-0418-
4ca5-80f8-5b6b4221ccee',
100 + '192.168.40.4' => '5c951242-56a9-
401a-8d74-5b7b4221ccee',
101 + '10.8.8.102' => '5c95396e-97c8-
49af-abb3-17e04221ccee'
102 + );
103
104 + // Include formatted sample data (this
file should be changed to your own database values and
user input)
105 + require_once('data.php');
106
107 + // Include dedicated IP checker
108 + require_once('checker.php');
109
110 + $checker = new DedicatedIpChecker();
111 + $verifiedIpList = $checker-
>verify($ipListToCheck, $userIps);
112
113 + ### Results
114 + echo '<pre>';
115
116 + foreach ($verifiedIpList as $ip =>
$userId) {
117 + echo 'IP: ' . $ip . "\n";
118 + echo 'User ID: ' . $userId .
"\n\n";
119 + }
120
121 + echo '</pre>';
122
123 + /* Results:

```

119
120 - IP: 172.16.88.10
121 - User ID: 5c953514-e9e8-4cca-954c-12334221ccee
122
123 - IP: 172.16.8.3
124 - User ID: 5c951454-0418-4ca5-80f8-5b6b4221ccee
125
126 - IP: 172.16.200.3
127 - User ID: 5c953514-e9e8-4cca-954c-12334221ccee
128
129 - IP: 172.16.201.4
130 - User ID: 5c952170-f908-45e9-af43-6bf64221ccee
131
132 - IP: 192.168.0.0
133 - User ID: 5c95396e-97c8-49af-abb3-17e04221ccee
134
135 - IP: 192.168.0.1
136 - User ID: 5c95396e-97c8-49af-abb3-17e04221ccee
137
138 - IP: 192.168.0.2
139 - User ID: 5c95396e-97c8-49af-abb3-17e04221ccee
140
141 - IP: 192.168.0.3
142 - User ID: 5c95396e-97c8-49af-abb3-17e04221ccee
143
144 - IP: 192.168.0.4
145 - User ID: 5c9518c1-0ad8-4e41-80a1-5bd54221ccee
146
147 - IP: 192.168.40.4
148 - User ID: 5c951242-56a9-401a-8d74-5b7b4221ccee
149
150 - IP: 192.168.49.4
151 - User ID: 5c952170-f908-45e9-af43-6bf64221ccee
152
153 - IP: 10.5.30.103
154 - User ID: 5c952170-f908-45e9-af43-6bf64221ccee
155
156 - IP: 10.5.30.105
157 - User ID: 5c952170-f908-45e9-af43-6bf64221ccee
158
159 - IP: 10.5.30.106
160 - User ID: 5c952170-f908-45e9-af43-6bf64221ccee
161
162 - IP: 10.5.31.108
163 - User ID: 5c95354f-43c8-4976-b69d-12334221ccee
164
165 - IP: 10.5.30.199
166 - User ID: 5c9518c1-0ad8-4e41-80a1-5bd54221ccee

124
125 + IP: 2001:db8:abcd:0008:847g:3e2:1088:8888
126 + User ID: 5c951454-0418-4ca5-80f8-5b6b4221ccee
127
128 + IP: 2001:db8:abcd:0008:0000:0000:ffff:ffff
129 + User ID: 5c953514-e9e8-4cca-954c-12334221ccee
130
131 + IP: 2001:db8:abcd:0008:0000:0000:8888:ffff
132 + User ID: 5c9518c1-0ad8-4e41-80a1-5bd54221ccee
133
134 + IP: 2001:db8:abcd:0008:0000:1234:8888:FFFF
135 + User ID: 5c953514-e9e8-4cca-954c-12334221ccee
136
137 + IP: 172.16.88.10
138 + User ID: 5c953514-e9e8-4cca-954c-12334221ccee
139
140 + IP: 172.16.8.3
141 + User ID: 5c951454-0418-4ca5-80f8-5b6b4221ccee
142
143 + IP: 172.16.200.3
144 + User ID: 5c953514-e9e8-4cca-954c-12334221ccee
145
146 + IP: 172.16.201.4
147 + User ID: 5c953514-e9e8-4cca-954c-12334221ccee
148
149 + IP: 192.168.0.0
150 + User ID: 5c951242-56a9-401a-8d74-5b7b4221ccee
151
152 + IP: 192.168.0.1
153 + User ID: 5c951242-56a9-401a-8d74-5b7b4221ccee
154
155 + IP: 192.168.0.2
156 + User ID: 5c95396e-97c8-49af-abb3-17e04221ccee
157
158 + IP: 192.168.0.3
159 + User ID: 5c95396e-97c8-49af-abb3-17e04221ccee
160
161 + IP: 192.168.0.4
162 + User ID: 5c95396e-97c8-49af-abb3-17e04221ccee
163
164 + IP: 192.168.40.4
165 + User ID: 5c951242-56a9-401a-8d74-5b7b4221ccee
166
167 + IP: 192.168.49.4
168 + User ID: 5c9518c1-0ad8-4e41-80a1-5bd54221ccee
169
170 + IP: 10.5.30.103
171 + User ID: 5c95396e-97c8-49af-abb3-

```

167
168 - IP: 10.5.89.44
169 - User ID: 5c952170-f908-45e9-af43-6bf64221ccee

170
171 - IP: 10.8.8.100
172 - User ID: 5c9518c1-0ad8-4e41-80a1-5bd54221ccee

173
174 - IP: 10.8.8.101
175 - User ID: 5c9518c1-0ad8-4e41-80a1-5bd54221ccee

176
177 - IP: 10.8.8.102
178 - User ID: 5c95396e-97c8-49af-abb3-17e04221ccee

179
180 - IP: 10.8.8.103
181 - User ID: 5c9518c1-0ad8-4e41-80a1-5bd54221ccee

182
183 - */

184
185 - ## Coming Soon
186 - * 1.0.1: IPv6 support

187
188 ## More From ParsonsBots
189 - https://github.com/parsonsbots
190 - https://gist.github.com/parsonsbots
191 - https://parsonsbots.com

17e04221ccee
173 + IP: 10.5.30.105
174 + User ID: 5c951454-0418-4ca5-80f8-5b6b4221ccee

175
176 + IP: 10.5.30.106
177 + User ID: 5c9518c1-0ad8-4e41-80a1-5bd54221ccee

178
179 + IP: 10.5.31.108
180 + User ID: 5c95354f-43c8-4976-b69d-12334221ccee

181
182 + IP: 10.5.30.199
183 + User ID: 5c9518c1-0ad8-4e41-80a1-5bd54221ccee

184
185 + IP: 10.5.89.44
186 + User ID: 5c952170-f908-45e9-af43-6bf64221ccee

187
188 + IP: 10.8.8.100
189 + User ID: 5c953514-e9e8-4cca-954c-12334221ccee

190
191 + IP: 10.8.8.101
192 + User ID: 5c953514-e9e8-4cca-954c-12334221ccee

193 +
194 + IP: 10.8.8.102
195 + User ID: 5c95396e-97c8-49af-abb3-17e04221ccee

196 +
197 + IP: 10.8.8.103
198 + User ID: 5c95396e-97c8-49af-abb3-17e04221ccee

199 +
200 + */

201
202 ## More From ParsonsBots
203 - https://github.com/parsonsbots
204 - https://gist.github.com/parsonsbots
205 - https://parsonsbots.com
206 + - https://eightomic.com

```

▼ 171 checker.php

```

... @@ -1,93 +1,171 @@
1 <?php
2 /**
3  * Dedicated IPv4 Address Checker
4  *
5  * Allow users to publicly verify dedicated IPv4
  addresses allocated to their unique, private account or
  user ID.
6  *
7  * @author Will Parsons
8  * @link https://parsonsbots.com

9 */

1 <?php
2 /**
3  + * Dedicated IP Address Checker
4  + *
5  + * Allow users to publicly verify dedicated IPv4 and
  IPv6 addresses allocated to their unique, private account
  or user ID.
6  + *
7  + * @copyright 2019 Will Parsons
8  + * @license https://github.com/parsonsbots/dedicated-
  ip-checker/blob/master/LICENSE // MIT License
9  + * @link https://parsonsbots.com
10 + * @link https://eightomic.com
11 */

```

```

10     class DedicatedIpChecker {
11
12     /**
13     - * Get user ID for an IPv4 address
14
15     *
16     - * @param string $ip IPv4 address
17
18     - * @param array $userIps User IPs
19     - * @param array $userIds User IDs
20     - * @return string $userId User ID for dedicated IP (if
21     user ID doesn't exist for IP, pick a UUID from existing
22     user IDs)
23
24     */
25     protected function _getUserId($ip, $userIps,
26     $userIds) {
27
28         if (!empty($userIps[$ip])) {
29             return $userIps[$ip];
30         }
31
32         $subnets = explode('.', $ip);
33         $key = ($subnets[0] != 0 ? $subnets[0] :
34         1) * ($subnets[3] != 0 ? $subnets[3] : 1) + $subnets[2] +
35         $subnets[1];
36
37         return ($userIds[$key]);
38     }
39
40     /**
41     - * Parse and filter IPv4 address list.
42
43     *
44     - * @param array $ips Unfiltered IPv4 address list
45     - * @return array $ips Filtered IPv4 address list

```

```

12     +
13     class DedicatedIpChecker {
14
15     /**
16     + * Get user ID for an IP address
17     + *
18     + * @param string $ip
19     + * @param array $userIps
20     + * @param array $userIds
21     + *
22     + * @return string $response User ID for dedicated IP (if
23     user ID doesn't exist for IP, pick a UUID from existing
24     user IDs)
25
26     */
27     protected function _getUserId($ip, $userIps,
28     $userIds) {
29
30         if (!empty($userIps[$ip])) {
31             return $userIps[$ip];
32         }
33
34         if (strpos($ip, ':') !== false) {
35             $characters =
36             'abcdefghijklmnopqrstuvwxyz';
37
38             $ipCharacters =
39             str_replace(array(0), '', strtolower($ip));
40
41             for ($i = 0; $i <
42             strlen($ipCharacters); $i++) {
43                 if (($numericValue =
44                 strpos($characters, $ipCharacters[$i])) !== false) {
45                     $ipCharacters[$i]
46                     = $numericValue + 1;
47                 }
48             }
49
50             $subnets =
51             array_filter(explode(':', $ipCharacters));
52             $key = array_sum($subnets) *
53             count($subnets);
54         } else {
55             $subnets = explode('.', $ip);
56             $key = ($subnets[0] != 0 ?
57             $subnets[0] : 1) * ($subnets[3] != 0 ? $subnets[3] : 1) +
58             $subnets[2] + $subnets[1];
59         }
60
61         $response = ($userIds[$key]);
62         return $response;
63     }
64
65     /**
66     + * Parse and filter IP address list
67     + *
68     + * @param mixed [array/string] $ips
69     + *
70     + * @return array $response

```

```

35  */
36  protected function _parseIps($ips = array()) {
37  -     $ips = implode("\n",
array_map(function($ip) {
38  -         return trim($ip, '.');

39  -     }, array_filter(preg_split("/[
(\r\n|\n|\r) @#$,[:_-]/", $ips))));
40  -     $ips = $this->_validateIps($ips);
41  -     return explode("\n", $ips);

42  }

43
44  /**
45  - * Validate IPv4 address list.

46  *
47  - * @param array $ips Filtered IPv4 address list
48  - * @return array $ips Validated IPv4 address list
49  */
50  -     protected function _validateIps($ips) {
51  -         $ips =
array_values(array_filter(explode("\n", $ips)));
52  -
53  -         foreach ($ips as $key => $ip) {
54  -             $splitIpSubnets =
array_map('trim', explode('.', trim($ip)));
55  -
56  -             if (count($splitIpSubnets) != 4) {
57  -                 unset($ips[$key]);
58  -                 continue;
59  -             }

60
61  -             foreach ($splitIpSubnets as
$splitIpSubnet) {
62  -                 if (
63  -
!is_numeric($splitIpSubnet) ||
64  -
strlen($splitIpSubnet) > 3 ||
65  -
$splitIpSubnet >
255 ||
66  -
$splitIpSubnet < 0
67  -             ) {
68  -                 unset($ips[$key]);
69  -                 continue;
70  -             }
71  -
72  -
73  -             $ips[$key] = $splitIpSubnets[0] .
'. ' . $splitIpSubnets[1] . ' ' . $splitIpSubnets[2] . ' '
. $splitIpSubnets[3];
74  -
75  -
76  -             return implode("\n", array_unique($ips));

```

```

56  */
57  protected function _parseIps($ips = array()) {
58  +     if (!is_array($ips)) {
59  +         $ips =
array_filter(preg_split("/[
(\r\n|\n|\r) <>()~
{}|'\"=?!*@#$,[:_-]/", $ips));
60  +     }

61  +
62  +         $response = $this->_validateIps($ips);
63  +         return $response;
64  +     }

65
66  /**
67  + * Validate IPv4 address
68  + *
69  + * @param string $ip
70  + *
71  + * @return mixed [boolean/string] $response

72  */
73  +     protected function _validateIpv4($ip) {
74  +         $response = false;

75  +
76  +
77  +         if (count($splitIpSubnets) === 4) {
78  +             foreach ($splitIpSubnets as
$splitIpSubnet) {
79  +                 if (
80  +
!is_numeric($splitIpSubnet) ||
81  +
strlen($splitIpSubnet) > 3 ||
82  +
$splitIpSubnet >
255 ||
83  +
$splitIpSubnet < 0
84  +             ) {
85  +                 return false;
86  +             }
87  +
88  +
89  +             $response = $ip;

90  +
91  +
92  +         return $response;
93  +     }
94  +
95  + /**
96  + * Validate IPv6 address
97  + *

```

```

98 + * @param string $ip
99 + *
100 + * @return mixed [boolean/string] $response
101 + */
102 +     protected function _validateIpv6($ip) {
103 +         $response = false;
104 +
105 +         if (strpos($ip, '::') !== false) {
106 +             $ip = str_replace(':',
107 +                 str_repeat('0000', 7 - (substr_count($ip, ':') - 1)) .
108 +                 ':', $ip);
109 +
110 +             $splitIpSubnets = explode(':', $ip);
111 +
112 +             if (count($splitIpSubnets) === 8) {
113 +                 foreach ($splitIpSubnets as
114 +                     $splitIpSubnet) {
115 +                     if (strlen($splitIpSubnet)
116 +                         > 4) {
117 +                         return false;
118 +                     }
119 +                 }
120 +
121 +                 $response = $ip;
122 +             }
123 +
124 +             return $response;
125 +         }
126 +
127 +         /**
128 +          * Validate IP address list
129 +          *
130 +          * @param array $ips
131 +          *
132 +          * @return array $response
133 +          */
134 +         protected function _validateIps($ips) {
135 +             foreach ($ips as $key => $ip) {
136 +                 if (
137 +                     empty($ip) ||
138 +                     strlen($ip) < 7 ||
139 +                     !($ip = trim($ip, '.')) ||
140 +                     !($ip = trim($ip, ':')) ||
141 +                     (
142 +                         strpos($ip, ':::')
143 +                         === false &&
144 +                         substr_count($ip,
145 +                             ':') > 4 &&
146 +                         ($ips[$key] =
147 +                             $this->_validateIpv6($ip)) === false
148 +                     ) ||
149 +                     (
150 +                         strpos($ip, ':')
151 +                         === false &&
152 +                         ($ips[$key] =
153 +                             $this->_validateIpv4($ip)) === false
154 +                     )
155 +                 ) {
156 +                     unset($ips[$key]);
157 +                 }
158 +             }
159 +         }
160 +     }

```

```

77     }
78
79     /**
80     * Verify dedicated IPs
81     *
82     - * @param array $ipv4List List of IPv4 addresses
      separated by new line
83     - * @param array $userIps List of dedicated IPv4
      addresses that belong to each unique user ID (IPv4 => UUID
      format)
84     - * @return $userIds
85
86     */
87     - public function verify($ipv4List, $userIps) {
88     -     $ipv4List = $this->_parseIps($ipv4List);
89     -     $userIds = array_values($userIps);
90
91     -     if (!empty($ipv4List)) {
92     -         $userIdsFormatted = false;
93
94     -         do {
95     -             } while ($userIdsFormatted ==
96     false);
97     -     }
98
99     -     foreach ($ipv4List as $key => $ip) {
100
101     -         $ips[$ip] = $this->_getUserId($ip,
102     $userIps, $userIds);
103     -         unset($ips[$key]);
104     -     }
105
106     -     return array_merge($ips, $userIps);
107
108     }
109
110 }

```

```

152 +         $response = implode("\n",
153 +         array_unique($ips));
154 +     +     return $response;
155 +     }
156
157     /**
158     * Verify dedicated IPs
159     *
160     + * @param array $ips
161     + * @param array $userIps
162
163     + * @return $response
164
165     */
166     + public function verify($ips, $userIps) {
167     +     $ips = $this->_parseIps($ips);
168     +     $userIds = array_values($userIps);
169
170     +     if (!empty($ips)) {
171     +         $userIdsFormatted = false;
172
173     +         do {
174     +             } while ($userIdsFormatted ==
175     false);
176     +     }
177
178     +     $ips = explode("\n", $ips);
179
180     +     foreach ($ips as $key => $ip) {
181     +         $ips[$ip] = $this->_getUserId($ip,
182     $userIps, $userIds);
183     +         unset($ips[$key]);
184     +     }
185
186     +     $response = array_merge($ips, $userIps);
187     +     return $response;
188
189     }
190
191 }

```

▼ 18 data.php

```

... @@ -1,12 +1,15 @@
1   <?php
2   /**
3   - * IPv4 list to check separated by new line
4   - * This list should be submitted through a public HTML
      form textarea field
5   *
6   - * Reserved IPv4 addresses below are used for demo
      purposes
7   *
8   */
9   -     $ipv4ListToCheck = implode("\n", array(
10
11     '172.16.88.10',
12     '172.16.8.3',
13     '172.16.200.3',

```

```

1   <?php
2   /**
3   + * IP list to check separated by new line
4   *
5   + * Reserved IP addresses below are used for demo purposes
6   *
7   */
8   +     $ipListToCheck = implode("\n", array(
9   +         '2001:db8:abcd:0008:847g:3e2:1088:8888',
10  +         '2001:db8:abcd:0008::ffff:ffff',
11  +         '2001:db8:abcd:0008:0000:0000:8888:ffff',
12  +         '2001:db8:abcd:0008::1234:8888:FFFF',
13  +         '172.16.88.10',
14  +         '172.16.8.3',
15  +         '172.16.200.3',

```



```

22         '10.5.30.105',
23         '10.5.30.106',
24         '10.5.31.108',
25 -         '_.10.5.30.199:80;*', // Invalid IP formats
    will be parsed
26         '10.5.89.44',
27         '10.8.8.100',
28         '10.8.8.101',
31     ));
32
33     /**
34 -    * List of dedicated IPv4 addresses that belong to each
    user (IPv4 => UUID format)
35     * Users should have access to see their own user ID
36     *
37 -    * Reserved IPv4 addresses below are used for demo
    purposes
38     *
39     */
40     $userIps = array(
41
42         '192.168.0.2' => '5c95396e-97c8-49af-abb3-17e04221ccee',
43         '192.168.0.3' => '5c95396e-97c8-49af-abb3-17e04221ccee',
44         '10.5.31.108' => '5c95354f-43c8-4976-b69d-12334221ccee',

```

```

25         '10.5.30.105',
26         '10.5.30.106',
27         '10.5.31.108',
28 +         '_.10.5.30.199;*', // Invalid IP formats
    will be parsed
29         '10.5.89.44',
30         '10.8.8.100',
31         '10.8.8.101',
34     ));
35
36     /**
37 +    * List of dedicated IP addresses that belong to each
    user (IP => UUID format)
38     * Users should have access to see their own user ID
39     *
40 +    * Reserved IP addresses below are used for demo purposes
41     *
42     */
43     $userIps = array(
44 +
45         '2001:db8:abcd:0008:0000:0000:8888:ffff' =>
46         '5c9518c1-0ad8-4e41-80a1-5bd54221ccee',
47         '192.168.0.2' => '5c95396e-97c8-49af-abb3-17e04221ccee',
48         '192.168.0.3' => '5c95396e-97c8-49af-abb3-17e04221ccee',
49         '10.5.31.108' => '5c95354f-43c8-4976-b69d-12334221ccee',

```

5 demo.php

```

... @@ -1,13 +1,12 @@
1     <?php
2         require_once('data.php');
3         require_once('checker.php');
4 -
5         $checker = new DedicatedIpChecker();
6 -         $verifiedIpv4List = $checker-
>verify($ipV4ListToCheck, $userIps);
7
8         echo '<pre>';
9
10 -        foreach ($verifiedIpv4List as $ip => $userId) {
11             echo 'IP: ' . $ip . "\n";
12             echo 'User ID: ' . $userId . "\n\n";
13         }

```

```

1     <?php
2         require_once('data.php');
3         require_once('checker.php');
4
5         $checker = new DedicatedIpChecker();
6 +         $verifiedIpList = $checker->verify($ipListToCheck,
    $userIps);
7
8         echo '<pre>';
9
10 +        foreach ($verifiedIpList as $ip => $userId) {
11             echo 'IP: ' . $ip . "\n";
12             echo 'User ID: ' . $userId . "\n\n";
13         }

```

0 comments on commit 01e8f12