




Update dynamic_reconfiguration.php

[Browse files](#) master parsonsbots committed on Mar 261 parent [aef39cd](#)commit [cf535d428d353a1e672c73e8d66b2dcf15ddf957](#) Showing 1 changed file with 295 additions and 298 deletions.

Unified

Split

▼ 593  dynamic_reconfiguration.php 

```
23         $this->sshPorts = $sshPorts;
24     }
25
```

```
26     /**
27      * Apply firewall rules
28      *
46         return;
47     }
48
```

```
23         $this->sshPorts = $sshPorts;
24     }
25
26     + /**
27     +  * Apply firewall
28     +  *
29     +  * @return boolean
30     +  */
31     +     protected function _applyFirewall() {
32     +         if
33     +         (file_exists('/scripts/pid/reconfigure.pid')) {
34     +             return false;
35     +         }
36     +         $gatewaysData =
37     +         json_decode(file_get_contents('/scripts/cache/gatewaysData
38     +         '), true);
39     +         if (empty($gatewaysData['data']
40     +         ['proxies'])) {
41     +             return false;
42     +         }
43     +         $overridePorts = array(
44     +             'http' => $this->processes['http']
45     +         );
46     +         $firewallRules = $this-
47     +         >_configureFirewallRules($gatewaysData, false,
48     +         $overridePorts);
49     +         $this->_applyFirewallRules($firewallRules,
50     +         'elastic');
51     +     }
52
53     + /**
54     +  * Apply firewall rules
55     +  *
69     +         return;
70     +     }
71
72     + /**
73     +  * Apply seamless processes reconfiguration
74     +  *
75     +  * @return boolean
76     +  */
77     +     protected function _applyReconfiguration() {
78     +         // Create writable log and cache
79     +         directories
80     +         $this->_createDirectories();
```

```

80 +
81 +           // Check for existing reconfiguration
process
82 +           if
(file_exists('/scripts/pid/reconfigure.pid')) {
83 +           $lastRan =
file_get_contents('/scripts/pid/reconfigure.pid');
84 +
85 +           // Start new reconfiguration
process if 15 minutes has passed
86 +           if ($lastRan < strtotime('-15
minutes')) {
87 +
unlink('/scripts/pid/reconfigure.pid');
88 +           } else {
89 +           return false;
90 +           }
91 +       }
92 +
93 +       $gatewaysJsonData = shell_exec("curl " .
$this->api . " --connect-timeout 30");
94 +       $gatewaysData =
json_decode($gatewaysJsonData, true);
95 +
96 +       // Log API error timestamp
97 +       if (
98 +           empty($gatewaysData['data']) ||
99 +           !is_dir('/etc/squid3')
100 +       ) {
101 +
file_put_contents('/scripts/errors/api-error-' . time(),
$gatewaysData);
102 +
unlink('/scripts/pid/reconfigure.pid');
103 +       return false;
104 +       }
105 +
106 +       // Require Squid and sysctl configurations
from API
107 +       if (
108 +           empty($gatewaysData['data']
['squid_conf']) ||
109 +           empty($gatewaysData['data']
['squid_redundant_conf']) ||
110 +           empty($gatewaysData['data']
['sysctl_conf'])
111 +       ) {
112 +
unlink('/scripts/pid/reconfigure.pid');
113 +       return false;
114 +       }
115 +
116 +       // Cache new ACLs from API
117 +
file_put_contents('/scripts/cache/gatewaysData',
$gatewaysJsonData);
118 +
119 +       // Create new reconfiguration process ID
120 +
file_put_contents('/scripts/pid/reconfigure.pid', time());
121 +
122 +       // Apply primary Squid config
123 +

```

```

file_put_contents('/etc/squid3/squid.conf',
$gatewaysData['data']['squid_conf']);
124 +
125 +         // TODO: Apply global redundant squid
configuration to all redundant processes
126 +         file_put_contents('/etc/squid3/squid-
redundant.conf', $gatewaysData['data']
['squid_redundant_conf']);
127 +
128 +         // Save and apply sysctl settings
129 +         file_put_contents('/etc/sysctl.conf',
$gatewaysData['data']['sysctl_conf']);
130 +         shell_exec('sysctl -p');
131 +
132 +         // Don't run reconfiguration if there
aren't any ACLs to apply to proxy IPs
133 +         $proxies = $gatewaysData['data']
['proxies'];
134 +
135 +         if (empty($proxies[0])) {
136 +
unlink('/scripts/pid/reconfigure.pid');
137 +         return false;
138 +         }
139 +
140 +         // Create new Squid user directories with
chunked sources and destinations
141 +         shell_exec('rm -rf /etc/squid3/users/');
142 +         shell_exec('mkdir -m 777
/etc/squid3/users/');
143 +
144 +         if (!empty($gatewaysData['data']
['files'])) {
145 +             foreach ($gatewaysData['data']
['files'] as $file) {
146 +                 shell_exec('mkdir -m 777 '
. str_replace(array('s.txt', 'd.txt'), '',
$file['path']));
147 +                 shell_exec('touch ' .
$file['path']);
148 +                 file_put_contents($file['path'], $file['contents']);
149 +             }
150 +         }
151 +
152 +         $firewallRules = $this->_configureFirewallRules($gatewaysData, true);
153 +
154 +         // Save Squid ACLs from API to file
155 +         shell_exec('rm
/etc/squid3/proxy_ip_acl.conf');
156 +         shell_exec('touch
/etc/squid3/proxy_ip_acl.conf');
157 +
file_put_contents('/etc/squid3/proxy_ip_acl.conf',
implode("\n", $gatewaysData['data']['acls']));
158 +
159 +         // Set proxy usernames and passwords using
htpasswd and basic_ncsa_auth for security
160 +         shell_exec('htpasswd -cb
/etc/squid3/passwords default default');
161 +         shell_exec('htpasswd -D
/etc/squid3/passwords default');

```

```

162 +
163 +         if (!empty($gatewaysData['data']
164 +             ['users'])) {
165 +             foreach ($gatewaysData['data']
166 +                 ['users'] as $username => $password) {
167 +                 shell_exec('htpasswd -b
168 + /etc/squid3/passwords ' . $username . ' ' . $password);
169 +             }
170 +         }
171 +
172 +         // Apply redundant firewall rules to begin
173 +         // seamless reconfiguration
174 +         $this->_applyFirewallRules($firewallRules,
175 +             'redundant');
176 +
177 +         // Reconfigure existing SOCKS processes
178 +         // first once redundant firewall is applied
179 +         if (
180 +             !empty($gatewaysData['data']
181 +                 ['socks']) &&
182 +             !empty($this->processes['socks']
183 +                 [0])
184 +         ) {
185 +             // SOCKS config and ACLs are in
186 +             // the same file (TODO: move squid configuration and ACLs to
187 +             // same file for consistency)
188 +             file_put_contents('/usr/local/etc/3proxy/3proxy.cfg',
189 +                 $gatewaysData['data']['socks']);
190 +
191 +             // Reconfigure main SOCKS instance
192 +             $this->_reconfigure(
193 +                 'socks',
194 +                 '3proxy',
195 +                 null,
196 +                 null,
197 +                 '3proxy.cfg',
198 +                 '/usr/local/etc/3proxy/3proxy.pid',
199 +                 10,
200 +                 20
201 +             );
202 +
203 +             // Reconfigure main HTTP instance
204 +             $this->_reconfigure(
205 +                 'http',
206 +                 'squid3',
207 +                 'squid3 -k reconfigure',
208 +                 'error:',
209 +                 'squid3',
210 +                 '/var/run/squid3.pid',
211 +                 0,
212 +                 75
213 +             );
214 +
215 +             $redundantProcesses = $this->
216 +                 >processes['http']; // TODO: chunk and reconfigure HTTP
217 +                 // and SOCKS processes simultaneously
218 +             unset($redundantProcesses[0]);
219 +             $redundantProcessChunks =
220 +                 array_chunk($redundantProcesses, 5, true);
221 +

```

```

209 +             foreach ($redundantProcessChunks as
210 +                 // Define active redundant process
211 +                 $activeRedundantProcesses =
212 +                 array_keys($redundantProcesses);
213 +                 // Define redundant process number
214 +                 range for reconfiguration
215 +                 reset($redundantProcessChunk);
216 +                 $redundantProcessStart =
217 +                 key($redundantProcessChunk);
218 +                 $redundantProcessEnd =
219 +                 key($redundantProcessChunk);
220 +                 $redundantProcessRange =
221 +                 range($redundantProcessStart, $redundantProcessEnd);
222 +                 foreach ($redundantProcessRange as
223 +                 $redundantProcess) {
224 +                 unset($activeRedundantProcesses[$redundantProcess - 1]);
225 +                 }
226 +                 // Get list of active redundant
227 +                 ports for firewall configuration
228 +                 $activeRedundantPorts = array();
229 +                 foreach ($redundantProcesses as
230 +                 $key => $redundantProcess) {
231 +                 if (in_array($key,
232 +                 $activeRedundantProcesses)) {
233 +                 $activeRedundantPorts = array_merge($activeRedundantPorts,
234 +                 $redundantProcesses[$key]);
235 +                 }
236 +                 }
237 +                 $overridePorts = array(
238 +                 'http' =>
239 +                 array_merge(array(
240 +                 '80',
241 +                 '8888',
242 +                 '55555'
243 +                 ), $activeRedundantPorts),
244 +                 'socks' => array(
245 +                 '1090'
246 +                 )
247 +                 );
248 +                 $firewallRules = $this->
249 +                 _configureFirewallRules($gatewaysData, false,
250 +                 $overridePorts);
251 +                 unlink('/scripts/iptables/iptables-redundant');
252 +                 $this->
253 +                 _applyFirewallRules($firewallRules, 'elastic');
254 +                 foreach ($redundantProcessRange as
255 +                 $redundantProcess) {
256 +                 // Reconfigure redundant
257 +                 HTTP instances
258 +                 $this->_reconfigure(

```

```

251 +                                     'http',
252 +                                     'squid3-redundant'
    . $redundantProcess,
253 +                                     'squid3-redundant'
    . $redundantProcess . ' -k reconfigure -f
    /etc/squid3/squid-redundant' . $redundantProcess .
    '.conf',
254 +                                     ' error:',
255 +                                     'squid3-redundant'
    . $redundantProcess,
256 +                                     '/var/run/squid-
    redundant' . $redundantProcess . '.pid',
257 +                                     0,
258 +                                     0
    );
259 +
260 +     }
261 +
262 +     // Fixed delay necessary to
    circumvent connection errors from varying downtime during
    reconfiguration with bulk ACLs
263 +     sleep(75);
264 + }
265 +
266 +     $overridePorts = array(
267 +         'http' => $this->processes['http']
268 +     );
269 +     $firewallRules = $this->
    _configureFirewallRules($gatewaysData, false,
    $overridePorts);
270 +     $this->_applyFirewallRules($firewallRules,
    'elastic');
271 +
272 +     if (!empty($gatewaysData['data']['socks-
    redundant'])) {
273 +         file_put_contents('/usr/local/etc/3proxy/3proxy.cfg',
    $gatewaysData['data']['socks']);
274 +
275 +         // Reconfigure redundant SOCKS
    instance
276 +         $this->_reconfigure(
277 +             'socks',
278 +             '3proxy-redundant',
279 +             null,
280 +             null,
281 +             '3proxy-redundant.cfg',
282 +             '/usr/local/etc/3proxy/3proxy-redundant.pid',
283 +             15,
284 +             0
285 +         );
286 +     }
287 +
288 +     $this->_applyFirewallRules($firewallRules,
    'elastic');
289 +
290 +     // Remove reconfiguration process ID
291 +     unlink('/scripts/pid/reconfigure.pid');
292 +
293 +     // Memory cleanup
294 +     gc_collect_cycles();
295 +
296 +     return true;

```

```

49  /**
50   * Configure firewall rules
51   *
170         // End DNAT load balancing for each
process
171     $rules[] = 'COMMIT';
172
173     // Chunk firewall rules to write to file
174     $rules = array_chunk($rules, 100);
175
176     return $rules;
177 }

297 +     }
298 +
299 + /**
300 +  * Check HTTP and SOCKS ports
301 +  *
302 +  * @param string $ip Proxy IP
303 +  * @param string $port Proxy port
304 +  * @param string $protocol Proxy protocol (http or socks)
305 +  * @param integer $integer Request timeout
306 +  *
307 +  * @return boolean $alive True if port is active, false
if refusing connections
308 +  */
309 +     protected function _checkPort($ip, $port,
$protocol, $timeout = 5) {
310 +         $alive = false;
311 +
312 +         switch ($protocol) {
313 +             case 'http':
314 +                 $response =
shell_exec('curl -I -s -x ' . $ip . ':' . $port . '
http://squid -v --connect-timeout ' . $timeout . ' --max-
time ' . $timeout);
315 +
316 +                 if ($this-
>_strpos(strtolower($response), array(
317 +                     '407 proxy',
318 +                     '403 forbidden',
319 +                     '503 ',
320 +                     'timed out '
321 +                 )) === false) {
322 +                     $alive = true;
323 +                 }
324 +
325 +                 break;
326 +             case 'socks':
327 +                 $socksResponse =
exec('curl --socks5-
hostname ' . $ip . ':' . $port . ' http://socks/ -v --
connect-timeout ' . $timeout . ' --max-time ' . $timeout .
' 2>&1', $socksResponse);
328 +                 $socksResponse =
end($socksResponse);
329 +                 $alive =
(strpos(strtolower($socksResponse), 'empty reply ') !==
false);
330 +                 break;
331 +             }
332 +
333 +             return $alive;
334 +         }
335 +
336 + /**
337 +  * Configure firewall rules
338 +  *
457         // End DNAT load balancing for each
process
458     $rules[] = 'COMMIT';
459
460     // Chunk firewall rules to write to file
461     $rules = array_chunk($rules, 100);
462
463     return $rules;

```

```

178 -
179 - /**
180 -  * Check HTTP and SOCKS ports
181 -  *
182 -  * @param string $ip Proxy IP
183 -  * @param string $port Proxy port
184 -  * @param string $port Proxy protocol (http or socks)
185 -  * @param integer $integer Request timeout
186 -  *
187 -  * @return boolean $alive True if port is active, false
188 -  * if refusing connections
189 -  */
190 -     protected function _checkPort($ip, $port,
191 - $protocol, $timeout = 5) {
192 -         $alive = false;
193 -
194 -         switch ($protocol) {
195 -             case 'http':
196 -                 $response =
197 - shell_exec('curl -I -s -x ' . $ip . ':' . $port . '
198 - http://squid -v --connect-timeout ' . $timeout . ' --max-
199 - time ' . $timeout);
200 -
201 -                 if ($this->_strposa(strtolower($response), array(
202 - '407 proxy',
203 - '403 forbidden',
204 - ' 503 ',
205 - ' timed out '
206 - )) === false) {
207 -                     $alive = true;
208 -                 }
209 -                 break;
210 -             case 'socks':
211 -                 exec('curl --socks5-
212 - hostname ' . $ip . ':' . $port . ' http://socks/ -v --
213 - connect-timeout ' . $timeout . ' --max-time ' . $timeout .
214 - ' 2>&1', $socksResponse);
215 -                 $socksResponse =
216 - end($socksResponse);
217 -                 $alive =
218 - (strpos(strtolower($socksResponse), 'empty reply ') !==
219 - false);
220 -                 break;
221 -         }
222 -         return $alive;
223 -     }
224 - }
225 - /**
226 -  */
227 - }
228 - }
229 -
230 - /**
231 -  * Apply firewall
232 -  *
233 -  * @return boolean
234 -  */
235 -     protected function _applyFirewall() {
236 -         if
237 - (file_exists('/scripts/pid/reconfigure.pid')) {
238 -             return false;

```

```

464     }
465
466     /**
474         }
475     }
476

```



```

335 -         }
336 -
337 -         $gatewaysData =
338 json_decode(file_get_contents('/scripts/cache/gatewaysData
339 '), true);
340 -
341 -         if (empty($gatewaysData['data']
342 ['proxies'])) {
343 -             return false;
344 -         }
345 -
346 -         $overridePorts = array(
347 -             'http' => $this->processes['http']
348         );
349 -         $firewallRules = $this->
350 _configureFirewallRules($gatewaysData, false,
351 $overridePorts);
352 -         $this->_applyFirewallRules($firewallRules,
353 'elastic');
354 -     }
355 -
356 - /**
357 -  * Apply seamless processes reconfiguration
358 -  *
359 -  * @return boolean
360 -  */
361 -     protected function _applyReconfiguration() {
362 -         // Create writable log and cache
363 directories
364 -         $this->_createDirectories();
365 -
366 -         // Check for existing reconfiguration
367 process
368 -         if
369 (file_exists('/scripts/pid/reconfigure.pid')) {
370 -             $lastRan =
371 file_get_contents('/scripts/pid/reconfigure.pid');
372 -
373 -             // Start new reconfiguration
374 process if 15 minutes has passed
375 -             if ($lastRan < strtotime('-15
376 minutes')) {
377 -
378 unlink('/scripts/pid/reconfigure.pid');
379 -             } else {
380 -                 return false;
381 -             }
382 -
383 -             $gatewaysJsonData = shell_exec("curl " .
384 $this->api . " --connect-timeout 30");
385 -             $gatewaysData =
386 json_decode($gatewaysJsonData, true);
387 -
388 -             // Log API error timestamp
389 -             if (
390 -                 empty($gatewaysData['data']) ||
391 -                 !is_dir('/etc/squid3')
392 -             ) {
393 -
394 file_put_contents('/scripts/errors/api-error-' . time(),
395 $gatewaysData);
396 -
397 -

```

```
unlink('/scripts/pid/reconfigure.pid');
381 -             return false;
382 -         }
383 -
384 -         // Require Squid and sysctl configurations
from API
385 -         if (
386 -             empty($gatewaysData['data']
['squid_conf']) ||
387 -             empty($gatewaysData['data']
['squid_redundant_conf']) ||
388 -             empty($gatewaysData['data']
['sysctl_conf'])
389 -         ) {
390 -
unlink('/scripts/pid/reconfigure.pid');
391 -             return false;
392 -         }
393 -
394 -         // Cache new ACLs from API
395 -
file_put_contents('/scripts/cache/gatewaysData',
$gatewaysJsonData);
396 -
397 -         // Create new reconfiguration process ID
398 -
file_put_contents('/scripts/pid/reconfigure.pid', time());
399 -
400 -         // Apply primary Squid config
401 -
file_put_contents('/etc/squid3/squid.conf',
$gatewaysData['data']['squid_conf']);
402 -
403 -         // TODO: Apply global redundant squid
configuration to all redundant processes
404 -         file_put_contents('/etc/squid3/squid-
redundant.conf', $gatewaysData['data']
['squid_redundant_conf']);
405 -
406 -         // Save and apply sysctl settings
407 -         file_put_contents('/etc/sysctl.conf',
$gatewaysData['data']['sysctl_conf']);
408 -         shell_exec('sysctl -p');
409 -
410 -         // Don't run reconfiguration if there
aren't any ACLs to apply to proxy IPs
411 -         $proxies = $gatewaysData['data']
['proxies'];
412 -
413 -         if (empty($proxies[0])) {
414 -
unlink('/scripts/pid/reconfigure.pid');
415 -             return false;
416 -         }
417 -
418 -         // Create new Squid user directories with
chunked sources and destinations
419 -         shell_exec('rm -rf /etc/squid3/users/');
420 -         shell_exec('mkdir -m 777
/etc/squid3/users/');
421 -
422 -         if (!empty($gatewaysData['data']
['files'])) {
```

```

423 -             foreach ($gatewaysData['data']
['files'] as $file) {
424 -                 shell_exec('mkdir -m 777 '
. str_replace(array('s.txt', 'd.txt'), '',
$file['path']));
425 -                 shell_exec('touch ' .
$file['path']);
426 -
file_put_contents($file['path'], $file['contents']);
427 -             }
428 -         }
429 -
430 -         $firewallRules = $this->
_configureFirewallRules($gatewaysData, true);
431 -
432 -         // Save Squid ACLs from API to file
433 -         shell_exec('rm
/etc/squid3/proxy_ip_acl.conf');
434 -         shell_exec('touch
/etc/squid3/proxy_ip_acl.conf');
435 -
file_put_contents('/etc/squid3/proxy_ip_acl.conf',
implode("\n", $gatewaysData['data']['acls']));
436 -
437 -         // Set proxy usernames and passwords using
htpasswd and basic_ncsa_auth for security
438 -         shell_exec('htpasswd -cb
/etc/squid3/passwords default default');
439 -         shell_exec('htpasswd -D
/etc/squid3/passwords default');
440 -
441 -         if (!empty($gatewaysData['data']
['users'])) {
442 -             foreach ($gatewaysData['data']
['users'] as $username => $password) {
443 -                 shell_exec('htpasswd -b
/etc/squid3/passwords ' . $username . ' ' . $password);
444 -             }
445 -         }
446 -
447 -         // Apply redundant firewall rules to begin
seamless reconfiguration
448 -         $this->_applyFirewallRules($firewallRules,
'redundant');
449 -
450 -         // Reconfigure existing SOCKS processes
first once redundant firewall is applied
451 -         if (
452 -             !empty($gatewaysData['data']
['socks']) &&
453 -             !empty($this->processes['socks']
[0])
454 -         ) {
455 -             // SOCKS config and ACLs are in
the same file (TODO: move squid configuration and ACLs to
same file for consistency)
456 -
file_put_contents('/usr/local/etc/3proxy/3proxy.cfg',
$gatewaysData['data']['socks']);
457 -
458 -             // Reconfigure main SOCKS instance
459 -             $this->_reconfigure(
460 -                 'socks',

```

```

461 -             '3proxy',
462 -             null,
463 -             null,
464 -             '3proxy.cfg',
465 -
466 -             '/usr/local/etc/3proxy/3proxy.pid',
467 -             10,
468 -             20
469 -         );
470 -     }
471 -
472 -     // Reconfigure main HTTP instance
473 -     $this->_reconfigure(
474 -         'http',
475 -         'squid3',
476 -         'squid3 -k reconfigure',
477 -         ' error:',
478 -         'squid3',
479 -         '/var/run/squid3.pid',
480 -         0,
481 -         70
482 -     );
483 -
484 -     $redundantProcesses = $this->
485 -     >processes['http']; // TODO: chunk and reconfigure HTTP
486 -     and SOCKS processes simultaneously
487 -     unset($redundantProcesses[0]);
488 -     $redundantProcessChunks =
489 -     array_chunk($redundantProcesses, 5, true);
490 -
491 -     foreach ($redundantProcessChunks as
492 -     $redundantProcessChunk) {
493 -         // Define active redundant process
494 -         numbers
495 -         $activeRedundantProcesses =
496 -         array_keys($redundantProcesses);
497 -
498 -         // Define redundant process number
499 -         range for reconfiguration
500 -         reset($redundantProcessChunk);
501 -         $redundantProcessStart =
502 -         key($redundantProcessChunk);
503 -         end($redundantProcessChunk);
504 -         $redundantProcessEnd =
505 -         key($redundantProcessChunk);
506 -         $redundantProcessRange =
507 -         range($redundantProcessStart, $redundantProcessEnd);
508 -
509 -         foreach ($redundantProcessRange as
510 -         $redundantProcess) {
511 -             unset($activeRedundantProcesses[$redundantProcess - 1]);
512 -         }
513 -
514 -         // Get list of active redundant
515 -         ports for firewall configuration
516 -         $activeRedundantPorts = array();
517 -
518 -         foreach ($redundantProcesses as
519 -         $key => $redundantProcess) {
520 -             if (in_array($key,
521 -             $activeRedundantProcesses)) {

```

```

$activeRedundantPorts = array_merge($activeRedundantPorts,
$redundantProcesses[$key]);
508 -         }
509 -     }
510 -
511 -     $overridePorts = array(
512 -         'http' =>
513         array_merge(array(
514             '80',
515             '8888',
516             '55555'
517         ), $activeRedundantPorts),
518         'socks' => array(
519             '1090'
520         )
521     );
522 -
523 -     $firewallRules = $this->
524     _configureFirewallRules($gatewaysData, false,
525     $overridePorts);
526 -
527 -     foreach ($redundantProcessRange as
528     $redundantProcess) {
529 -         // Reconfigure redundant
530     HTTP instances
531 -         $this->_reconfigure(
532             'http',
533             'squid3-redundant'
534             . $redundantProcess,
535             'squid3-redundant'
536             . $redundantProcess . ' -k reconfigure -f
537             /etc/squid3/squid-redundant' . $redundantProcess .
538             '.conf',
539             ' error:',
540             'squid3-redundant'
541             . $redundantProcess,
542             '/var/run/squid-
543     redundant' . $redundantProcess . '.pid',
544             0,
545             0
546         );
547 -     }
548 -
549 -     // Fixed delay necessary to
550     circumvent connection errors from varying downtime during
    reconfiguration with bulk ACLs
551 -     sleep(75);
552 - }
553 -
554 -     $overridePorts = array(
555 -         'http' => $this->processes['http']
556     );
557 -
558 -     $firewallRules = $this->
559     _configureFirewallRules($gatewaysData, false,
560     $overridePorts);
561 -
562 -     $this->_applyFirewallRules($firewallRules,
563     'elastic');
564 -
565 -     if (!empty($gatewaysData['data']['socks-

```

<pre> 551 redundant')) { 552 - 553 - // Reconfigure redundant SOCKS 554 instance 555 - \$this->_reconfigure(556 - 'socks', 557 - '3proxy-redundant', 558 - null, 559 - null, 560 - '3proxy-redundant.cfg', 561 - 15, 562 - 0 563 -); 564 - } 565 - 566 \$this->_applyFirewallRules(\$firewallRules, 567 'elastic'); 568 - 569 // Remove reconfiguration process ID 570 unlink('/scripts/pid/reconfigure.pid'); 571 - 572 // Memory cleanup 573 gc_collect_cycles(); 574 - 575 return true; 576 } 577 /** 578 * Initiate processes 579 * 580 * @param string \$process Process name 581 * 582 * @return boolean \$status 583 */ 584 - public function start(\$process) { 585 - 586 date_default_timezone_set('America/Los_Angeles'); 587 - 588 - switch (\$process) { 589 - case 'apply_firewall': 590 - \$status = \$this->_applyFirewall(); 591 - break; 592 - 593 * 594 * @return boolean True if match is found, false if no 595 match 596 */ 597 - protected function strposa(\$haystack, \$needles, 598 \$offset = 0) { 599 - 600 if (!is_array(\$needles)) { 601 - \$needles = array(\$needles); 602 - }; </pre>	<pre> 577 /** 578 * Initiate processes 579 * 580 + * @param string \$processName Process name 581 * 582 * @return boolean \$status 583 */ 584 + public function start(\$processName) { 585 + 586 + switch (\$processName) { 587 - 588 case 'apply_firewall': 589 - \$status = \$this->_applyFirewall(); 590 - break; 591 - 592 * 593 * @return boolean True if match is found, false if no 594 match 595 */ 596 + protected function strposa(\$haystack, \$needles, 597 \$offset = 0) { 598 - 599 if (!is_array(\$needles)) { 600 - \$needles = array(\$needles); 601 - }; </pre>
---	--

0 comments on commit `cf535d4`