parsonsbots / dynamic-proxy-node-reconfiguration

Update dynamic_reconfiguration.php

master

Browse files

parsonsbots committed on Mar 26 — 1 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 (file_exists('/scripts/pid/reconfigure.pid')) {
33  +                      return false;
34  +              }
35  +
36  +              $gatewaysData = json_decode(file_get_contents('/scripts/cache/gatewaysData'), true);
37  +
38  +              if (empty($gatewaysData['data']['proxies'])) {
39  +                      return false;
40  +              }
41  +
42  +              $overridePorts = array(
43  +                      'http' => $this->processes['http']
44  +              );
45  +              $firewallRules = $this->_configureFirewallRules($gatewaysData, false, $overridePorts);
46  +              $this->_applyFirewallRules($firewallRules, 'elastic');
47  +      }
48  +
49    /**
50     * Apply firewall rules
51     *
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 directories
79  +              $this->_createDirectories();
```

```
+
+               // Check for existing reconfiguration process
+               if (file_exists('/scripts/pid/reconfigure.pid')) {
+                       $lastRan = file_get_contents('/scripts/pid/reconfigure.pid');
+
+                       // Start new reconfiguration process if 15 minutes has passed
+                       if ($lastRan < strtotime('-15 minutes')) {
+                               unlink('/scripts/pid/reconfigure.pid');
+                       } else {
+                               return false;
+                       }
+               }
+
+               $gatewaysJsonData = shell_exec("curl " . $this->api . " --connect-timeout 30");
+               $gatewaysData = json_decode($gatewaysJsonData, true);
+
+               // Log API error timestamp
+               if (
+                       empty($gatewaysData['data']) ||
+                       !is_dir('/etc/squid3')
+               ) {
+                       file_put_contents('/scripts/errors/api-error-' . time(), $gatewaysData);
+                       unlink('/scripts/pid/reconfigure.pid');
+                       return false;
+               }
+
+               // Require Squid and sysctl configurations from API
+               if (
+                       empty($gatewaysData['data']['squid_conf']) ||
+                       empty($gatewaysData['data']['squid_redundant_conf']) ||
+                       empty($gatewaysData['data']['sysctl_conf'])
+               ) {
+                       unlink('/scripts/pid/reconfigure.pid');
+                       return false;
+               }
+
+               // Cache new ACLs from API
+               file_put_contents('/scripts/cache/gatewaysData', $gatewaysJsonData);
+
+               // Create new reconfiguration process ID
+               file_put_contents('/scripts/pid/reconfigure.pid', time());
+
+               // Apply primary Squid config
+
```

```
file_put_contents('/etc/squid3/squid.conf', $gatewaysData['data']['squid_conf']);
+
+               // TODO: Apply global redundant squid configuration to all redundant processes
+               file_put_contents('/etc/squid3/squid-redundant.conf', $gatewaysData['data']['squid_redundant_conf']);
+
+               // Save and apply sysctl settings
+               file_put_contents('/etc/sysctl.conf', $gatewaysData['data']['sysctl_conf']);
+               shell_exec('sysctl -p');
+
+               // Don't run reconfiguration if there aren't any ACLs to apply to proxy IPs
+               $proxies = $gatewaysData['data']['proxies'];
+
+               if (empty($proxies[0])) {
+                       unlink('/scripts/pid/reconfigure.pid');
+                       return false;
+               }
+
+               // Create new Squid user directories with chunked sources and destinations
+               shell_exec('rm -rf /etc/squid3/users/');
+               shell_exec('mkdir -m 777 /etc/squid3/users/');
+
+               if (!empty($gatewaysData['data']['files'])) {
+                       foreach ($gatewaysData['data']['files'] as $file) {
+                               shell_exec('mkdir -m 777 ' . str_replace(array('s.txt', 'd.txt'), '', $file['path']));
+                               shell_exec('touch ' . $file['path']);
+                               file_put_contents($file['path'], $file['contents']);
+                       }
+               }
+
+               $firewallRules = $this->_configureFirewallRules($gatewaysData, true);
+
+               // Save Squid ACLs from API to file
+               shell_exec('rm /etc/squid3/proxy_ip_acl.conf');
+               shell_exec('touch /etc/squid3/proxy_ip_acl.conf');
+               file_put_contents('/etc/squid3/proxy_ip_acl.conf', implode("\n", $gatewaysData['data']['acls']));
+
+               // Set proxy usernames and passwords using htpasswd and basic_ncsa_auth for security
+               shell_exec('htpasswd -cb /etc/squid3/passwords default default');
+               shell_exec('htpasswd -D /etc/squid3/passwords default');
```

```
+
+               if (!empty($gatewaysData['data']['users'])) {
+                       foreach ($gatewaysData['data']['users'] as $username => $password) {
+                               shell_exec('htpasswd -b /etc/squid3/passwords ' . $username . ' ' . $password);
+                       }
+               }
+
+               // Apply redundant firewall rules to begin seamless reconfiguration
+               $this->_applyFirewallRules($firewallRules, 'redundant');
+
+               // Reconfigure existing SOCKS processes first once redundant firewall is applied
+               if (
+                       !empty($gatewaysData['data']['socks']) &&
+                       !empty($this->processes['socks'][0])
+               ) {
+                       // SOCKS config and ACLs are in the same file (TODO: move squid configuration and ACLs to same file for consistency)
+                       file_put_contents('/usr/local/etc/3proxy/3proxy.cfg', $gatewaysData['data']['socks']);
+
+                       // Reconfigure main SOCKS instance
+                       $this->_reconfigure(
+                               'socks',
+                               '3proxy',
+                               null,
+                               null,
+                               '3proxy.cfg',
+                               '/usr/local/etc/3proxy/3proxy.pid',
+                               10,
+                               20
+                       );
+               }
+
+               // Reconfigure main HTTP instance
+               $this->_reconfigure(
+                       'http',
+                       'squid3',
+                       'squid3 -k reconfigure',
+                       ' error:',
+                       'squid3',
+                       '/var/run/squid3.pid',
+                       0,
+                       75
+               );
+
+               $redundantProcesses = $this->processes['http']; // TODO: chunk and reconfigure HTTP and SOCKS processes simultaneously
+               unset($redundantProcesses[0]);
+               $redundantProcessChunks = array_chunk($redundantProcesses, 5, true);
+
```

```
+               foreach ($redundantProcessChunks as $redundantProcessChunk) {
+                       // Define active redundant process numbers
+                       $activeRedundantProcesses = array_keys($redundantProcesses);
+
+                       // Define redundant process number range for reconfiguration
+                       reset($redundantProcessChunk);
+                       $redundantProcessStart = key($redundantProcessChunk);
+                       end($redundantProcessChunk);
+                       $redundantProcessEnd = key($redundantProcessChunk);
+                       $redundantProcessRange = range($redundantProcessStart, $redundantProcessEnd);
+
+                       foreach ($redundantProcessRange as $redundantProcess) {
+                               unset($activeRedundantProcesses[$redundantProcess - 1]);
+                       }
+
+                       // Get list of active redundant ports for firewall configuration
+                       $activeRedundantPorts = array();
+
+                       foreach ($redundantProcesses as $key => $redundantProcess) {
+                               if (in_array($key, $activeRedundantProcesses)) {
+                                       $activeRedundantPorts = array_merge($activeRedundantPorts, $redundantProcesses[$key]);
+                               }
+                       }
+
+                       $overridePorts = array(
+                               'http' => array_merge(array(
+                                       '80',
+                                       '8888',
+                                       '55555'
+                               ), $activeRedundantPorts),
+                               'socks' => array(
+                                       '1090'
+                               )
+                       );
+
+                       $firewallRules = $this->_configureFirewallRules($gatewaysData, false, $overridePorts);
+                       unlink('/scripts/iptables/iptables-redundant');
+                       $this->_applyFirewallRules($firewallRules, 'elastic');
+
+                       foreach ($redundantProcessRange as $redundantProcess) {
+                               // Reconfigure redundant HTTP instances
+                               $this->_reconfigure(
```

```
+                                        'http',
+                                        'squid3-redundant' . $redundantProcess,
+                                        'squid3-redundant' . $redundantProcess . ' -k reconfigure -f /etc/squid3/squid-redundant' . $redundantProcess . '.conf',
+                                        ' error:',
+                                        'squid3-redundant' . $redundantProcess,
+                                        '/var/run/squid-redundant' . $redundantProcess . '.pid',
+                                        0,
+                                        0
+                                );
+                        }
+
+                        // Fixed delay necessary to circumvent connection errors from varying downtime during reconfiguration with bulk ACLs
+                        sleep(75);
+                }
+
+                $overridePorts = array(
+                        'http' => $this->processes['http']
+                );
+                $firewallRules = $this->_configureFirewallRules($gatewaysData, false, $overridePorts);
+                $this->_applyFirewallRules($firewallRules, 'elastic');
+
+                if (!empty($gatewaysData['data']['socks-redundant'])) {
+                        file_put_contents('/usr/local/etc/3proxy/3proxy.cfg', $gatewaysData['data']['socks']);
+
+                        // Reconfigure redundant SOCKS instance
+                        $this->_reconfigure(
+                                'socks',
+                                '3proxy-redundant',
+                                null,
+                                null,
+                                '3proxy-redundant.cfg',
+                                '/usr/local/etc/3proxy/3proxy-redundant.pid',
+                                15,
+                                0
+                        );
+                }
+
+                $this->_applyFirewallRules($firewallRules, 'elastic');
+
+                // Remove reconfiguration process ID
+                unlink('/scripts/pid/reconfigure.pid');
+
+                // Memory cleanup
+                gc_collect_cycles();
+
+                return true;
```

```
+       }
+
+ /**
+  * Check HTTP and SOCKS ports
+  *
+  * @param string $ip Proxy IP
+  * @param string $port Proxy port
+  * @param string $port Proxy protocol (http or socks)
+  * @param integer $integer Request timeout
+  *
+  * @return boolean $alive True if port is active, false if refusing connections
+  */
+       protected function _checkPort($ip, $port, $protocol, $timeout = 5) {
+               $alive = false;
+
+               switch ($protocol) {
+                       case 'http':
+                               $response = shell_exec('curl -I -s -x ' . $ip . ':' . $port . ' http://squid -v --connect-timeout ' . $timeout . ' --max-time ' . $timeout);
+
+                               if ($this->_strposa(strtolower($response), array(
+                                       '407 proxy',
+                                       '403 forbidden',
+                                       ' 503 ',
+                                       ' timed out '
+                               )) === false) {
+                                       $alive = true;
+                               }
+
+                               break;
+                       case 'socks':
+                               exec('curl --socks5-hostname ' . $ip . ':' . $port . ' http://socks/ -v --connect-timeout ' . $timeout . ' --max-time ' . $timeout . ' 2>&1', $socksResponse);
+                               $socksResponse = end($socksResponse);
+                               $alive = (strpos(strtolower($socksResponse), 'empty reply ') !== false);
+                               break;
+               }
+
+               return $alive;
+       }
+
```

```
  /**
   * Configure firewall rules
   *
                // End DNAT load balancing for each process
                $rules[] = 'COMMIT';

-               // Chunk firewall rules to write to file
-               $rules = array_chunk($rules, 100);
-
-               return $rules;
-       }
```

```
  /**
   * Configure firewall rules
   *
                // End DNAT load balancing for each process
                $rules[] = 'COMMIT';

+               // Chunk firewall rules to write to file
+               $rules = array_chunk($rules, 100);
+
+               return $rules;
```

```
-
- /**
-  * Check HTTP and SOCKS ports
-  *
-  * @param string $ip Proxy IP
-  * @param string $port Proxy port
-  * @param string $port Proxy protocol (http or socks)
-  * @param integer $integer Request timeout
-  *
-  * @return boolean $alive True if port is active, false if refusing connections
-  */
-       protected function _checkPort($ip, $port, $protocol, $timeout = 5) {
-               $alive = false;
-
-               switch ($protocol) {
-                       case 'http':
-                               $response = shell_exec('curl -I -s -x ' . $ip . ':' . $port . ' http://squid -v --connect-timeout ' . $timeout . ' --max-time ' . $timeout);
-
-                               if ($this->_strposa(strtolower($response), array(
-                                       '407 proxy',
-                                       '403 forbidden',
-                                       ' 503 ',
-                                       ' timed out '
-                               )) === false) {
-                                       $alive = true;
-                               }
-
-                               break;
-                       case 'socks':
-                               exec('curl --socks5-hostname ' . $ip . ':' . $port . ' http://socks/ -v --connect-timeout ' . $timeout . ' --max-time ' . $timeout . ' 2>&1', $socksResponse);
-                               $socksResponse = end($socksResponse);
-                               $alive = (strpos(strtolower($socksResponse), 'empty reply ') !== false);
-                               break;
-               }
-
-               return $alive;
        }

  /**
               }
       }

- /**
-  * Apply firewall
-  *
-  * @return boolean
-  */
-       protected function _applyFirewall() {
-               if (file_exists('/scripts/pid/reconfigure.pid')) {
-                       return false;
```

```
        }

/**
                }
        }

/**
        }
```

```
-               }
-
-               $gatewaysData = json_decode(file_get_contents('/scripts/cache/gatewaysData'), true);
-
-               if (empty($gatewaysData['data']['proxies'])) {
-                       return false;
-               }
-
-               $overridePorts = array(
-                       'http' => $this->processes['http']
-               );
-               $firewallRules = $this->_configureFirewallRules($gatewaysData, false, $overridePorts);
-               $this->_applyFirewallRules($firewallRules, 'elastic');
-       }
-
- /**
-  * Apply seamless processes reconfiguration
-  *
-  * @return boolean
-  */
-       protected function _applyReconfiguration() {
-               // Create writable log and cache directories
-               $this->_createDirectories();
-
-               // Check for existing reconfiguration process
-               if (file_exists('/scripts/pid/reconfigure.pid')) {
-                       $lastRan = file_get_contents('/scripts/pid/reconfigure.pid');
-
-                       // Start new reconfiguration process if 15 minutes has passed
-                       if ($lastRan < strtotime('-15 minutes')) {
-                               unlink('/scripts/pid/reconfigure.pid');
-                       } else {
-                               return false;
-                       }
-               }
-
-               $gatewaysJsonData = shell_exec("curl " . $this->api . " --connect-timeout 30");
-               $gatewaysData = json_decode($gatewaysJsonData, true);
-
-               // Log API error timestamp
-               if (
-                       empty($gatewaysData['data']) ||
-                       !is_dir('/etc/squid3')
-               ) {
-                       file_put_contents('/scripts/errors/api-error-' . time(), $gatewaysData);
-
```

```
unlink('/scripts/pid/reconfigure.pid');
-                       return false;
-               }
-
-               // Require Squid and sysctl configurations from API
-               if (
-                       empty($gatewaysData['data']['squid_conf']) ||
-                       empty($gatewaysData['data']['squid_redundant_conf']) ||
-                       empty($gatewaysData['data']['sysctl_conf'])
-               ) {
-                       unlink('/scripts/pid/reconfigure.pid');
-                       return false;
-               }
-
-               // Cache new ACLs from API
-               file_put_contents('/scripts/cache/gatewaysData', $gatewaysJsonData);
-
-               // Create new reconfiguration process ID
-               file_put_contents('/scripts/pid/reconfigure.pid', time());
-
-               // Apply primary Squid config
-               file_put_contents('/etc/squid3/squid.conf', $gatewaysData['data']['squid_conf']);
-
-               // TODO: Apply global redundant squid configuration to all redundant processes
-               file_put_contents('/etc/squid3/squid-redundant.conf', $gatewaysData['data']['squid_redundant_conf']);
-
-               // Save and apply sysctl settings
-               file_put_contents('/etc/sysctl.conf', $gatewaysData['data']['sysctl_conf']);
-               shell_exec('sysctl -p');
-
-               // Don't run reconfiguration if there aren't any ACLs to apply to proxy IPs
-               $proxies = $gatewaysData['data']['proxies'];
-
-               if (empty($proxies[0])) {
-                       unlink('/scripts/pid/reconfigure.pid');
-                       return false;
-               }
-
-               // Create new Squid user directories with chunked sources and destinations
-               shell_exec('rm -rf /etc/squid3/users/');
-               shell_exec('mkdir -m 777 /etc/squid3/users/');
-
-               if (!empty($gatewaysData['data']['files'])) {
```

```
-                       foreach ($gatewaysData['data']['files'] as $file) {
-                               shell_exec('mkdir -m 777 ' . str_replace(array('s.txt', 'd.txt'), '', $file['path']));
-                               shell_exec('touch ' . $file['path']);
-                               file_put_contents($file['path'], $file['contents']);
-                       }
-               }
-
-               $firewallRules = $this->_configureFirewallRules($gatewaysData, true);
-
-               // Save Squid ACLs from API to file
-               shell_exec('rm /etc/squid3/proxy_ip_acl.conf');
-               shell_exec('touch /etc/squid3/proxy_ip_acl.conf');
-               file_put_contents('/etc/squid3/proxy_ip_acl.conf', implode("\n", $gatewaysData['data']['acls']));
-
-               // Set proxy usernames and passwords using htpasswd and basic_ncsa_auth for security
-               shell_exec('htpasswd -cb /etc/squid3/passwords default default');
-               shell_exec('htpasswd -D /etc/squid3/passwords default');
-
-               if (!empty($gatewaysData['data']['users'])) {
-                       foreach ($gatewaysData['data']['users'] as $username => $password) {
-                               shell_exec('htpasswd -b /etc/squid3/passwords ' . $username . ' ' . $password);
-                       }
-               }
-
-               // Apply redundant firewall rules to begin seamless reconfiguration
-               $this->_applyFirewallRules($firewallRules, 'redundant');
-
-               // Reconfigure existing SOCKS processes first once redundant firewall is applied
-               if (
-                       !empty($gatewaysData['data']['socks']) &&
-                       !empty($this->processes['socks'][0])
-               ) {
-                       // SOCKS config and ACLs are in the same file (TODO: move squid configuration and ACLs to same file for consistency)
-                       file_put_contents('/usr/local/etc/3proxy/3proxy.cfg', $gatewaysData['data']['socks']);
-
-                       // Reconfigure main SOCKS instance
-                       $this->_reconfigure(
-                               'socks',
```

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

```
$activeRedundantPorts = array_merge($activeRedundantPorts, $redundantProcesses[$key]);
-                               }
-                       }
-
-                       $overridePorts = array(
-                               'http' => array_merge(array(
-                                       '80',
-                                       '8888',
-                                       '55555'
-                               ), $activeRedundantPorts),
-                               'socks' => array(
-                                       '1090'
-                               )
-                       );
-
-                       $firewallRules = $this->_configureFirewallRules($gatewaysData, false, $overridePorts);
-
unlink('/scripts/iptables/iptables-redundant');
-                       $this->_applyFirewallRules($firewallRules, 'elastic');
-
-                       foreach ($redundantProcessRange as $redundantProcess) {
-                               // Reconfigure redundant HTTP instances
-                               $this->_reconfigure(
-                                       'http',
-                                       'squid3-redundant' . $redundantProcess,
-                                       'squid3-redundant' . $redundantProcess . ' -k reconfigure -f /etc/squid3/squid-redundant' . $redundantProcess . '.conf',
-                                       ' error:',
-                                       'squid3-redundant' . $redundantProcess,
-                                       '/var/run/squid-redundant' . $redundantProcess . '.pid',
-                                       0,
-                                       0
-                               );
-                       }
-
-                       // Fixed delay necessary to circumvent connection errors from varying downtime during reconfiguration with bulk ACLs
-                       sleep(75);
-               }
-
-               $overridePorts = array(
-                       'http' => $this->processes['http']
-               );
-               $firewallRules = $this->_configureFirewallRules($gatewaysData, false, $overridePorts);
-               $this->_applyFirewallRules($firewallRules, 'elastic');
-
-               if (!empty($gatewaysData['data']['socks-
```

```
redundant'])) {
-
file_put_contents('/usr/local/etc/3proxy/3proxy.cfg',
$gatewaysData['data']['socks']);
-
-                       // Reconfigure redundant SOCKS
instance
-                       $this->_reconfigure(
-                               'socks',
-                               '3proxy-redundant',
-                               null,
-                               null,
-                               '3proxy-redundant.cfg',
-
'/usr/local/etc/3proxy/3proxy-redundant.pid',
-                               15,
-                               0
-                       );
-               }
-
-               $this->_applyFirewallRules($firewallRules,
'elastic');
-
-               // Remove reconfiguration process ID
-               unlink('/scripts/pid/reconfigure.pid');
-
-               // Memory cleanup
-               gc_collect_cycles();
-
-               return true;
-       }
-
  /**
   * Initiate processes
   *
-  * @param string $process Process name
+  * @param string $processName Process name
   *
   * @return boolean $status
   */
-       public function start($process) {
-
date_default_timezone_set('America/Los_Angeles');
-               ini_set('memory_limit', -1);
-
-               switch ($process) {
                        case 'apply_firewall':
                                $status = $this-
>_applyFirewall();
                                break;
   *
   * @return boolean True if match is found, false if no
match
   */
-       protected function strposa($haystack, $needles,
$offset = 0) {
                if (!is_array($needles)) {
                        $needles = array($needles);
                };
```

```
  /**
   * Initiate processes
   *
+  * @param string $processName Process name
   *
   * @return boolean $status
   */
+       public function start($processName) {
+               switch ($processName) {
                        case 'apply_firewall':
                                $status = $this-
>_applyFirewall();
                                break;
   *
   * @return boolean True if match is found, false if no
match
   */
+       protected function _strposa($haystack, $needles,
$offset = 0) {
                if (!is_array($needles)) {
                        $needles = array($needles);
                };
```

0 comments on commit `cf535d4`