parsonsbots / **dynamic-proxy-node-reconfiguration**

Update dynamic_reconfiguration.php

master 1.0.0

Browse files

parsonsbots committed on Mar 26

1 parent 2e94435 commit d09e8c2c297e68c4cff0970c6debb2a2dd5f7189

Showing 1 changed file with **246 additions** and **285 deletions**.

Unified Split

531 🟩🟩🟥🟥⬜ dynamic_reconfiguration.php

```
        public $api;
        public $processes;

        public $sshPorts;

        public function __construct($api, $processes, $protocols) {
                $this->api = $api;
                $this->processes = $processes;

                $this->sshPorts = $sshPorts;
        }

  /**
   * Configure firewall rules
   *
   * @param array $gatewaysData Data from gateway API response
```

```

        public $api;
        public $processes;
+       public $shell;
        public $sshPorts;

        public function __construct($api, $processes, $protocols) {
                $this->api = $api;
                $this->processes = $processes;
+               $this->shell = $shell;
                $this->sshPorts = $sshPorts;
        }

+ /**
+  * Apply firewall rules
+  *
+  * @param array $firewallRules Firewall rules
+  * @param string $ruleSet Firewall rule set
+  *
+  * @return
+  */
+       protected function _applyFirewallRules($firewallRules, $ruleSet) {
+               unlink('/scripts/iptables/iptables-' . ruleSet);
+               touch('/scripts/iptables/iptables-' . ruleSet);
+
+               foreach ($firewallRules as $ruleChunk) {
+                       $saveRules = implode("\n", $ruleChunk);
+                       shell_exec('echo "' . $saveRules . '" >> /scripts/iptables/iptables-' . ruleSet);
+               }
+
+               shell_exec('iptables-restore < /scripts/iptables/iptables-' . ruleSet);
+               sleep(5);
+               shell_exec('iptables-save');
+               return;
+       }
+
  /**
   * Configure firewall rules
   *
   * @param array $gatewaysData Data from gateway API response
   *
-  * @return $rules
   */
-     protected function _configureFirewallRules($gatewaysData) {
          // Begin input filter rules
          $rules = array(
              '*filter',
              is_array($this->processes)
          ) {
              foreach ($this->processes as $protocol => $processes) {
                  foreach ($processes as $ports) {
                      $basePorts = array_merge($ports, $basePorts);
                  }

                  $ports = array_unique($basePorts);

                  // Remove ports from routing list that are refusing connections
                  foreach ($ports as $key => $port) {
                      if ($this->_checkPort($gatewaysData['data']['proxies'][0], $port, $protocol) === false) {
                          unset($ports[$key]);
                      }
                  }

-                 // Retrieve list of primary ports for
                  $primaryPorts = $processes[0];

-                 // Prevent listening ports from overloading without changing destination IP using DNAT port load balancing
```

```
+  * @param boolean $redundant Route to redundant process ports only if true, all ports if false
+  * @param array $ports Override default base ports
   *
+  * @return array $rules Firewall rules
   */
+     protected function _configureFirewallRules($gatewaysData, $redundant = false, $overridePorts = array()) {
          // Begin input filter rules
          $rules = array(
              '*filter',
              is_array($this->processes)
          ) {
              foreach ($this->processes as $protocol => $processes) {
+                 if ($redundant) {
+                     unset($processes[0]);
+                 }
+
                  foreach ($processes as $ports) {
                      $basePorts = array_merge($ports, $basePorts);
                  }

                  $ports = array_unique($basePorts);

+                 if (!empty($overridePorts)) {
+                     if (empty($overridePorts[$protocol])) {
+                         continue;
+                     }
+
+                     $ports = $overridePorts[$protocol];
+                 }
+
                  // Remove ports from routing list that are refusing connections
                  foreach ($ports as $key => $port) {
                      if ($this->_checkPort($gatewaysData['data']['proxies'][0], $port, $protocol) === false) {
                          unset($ports[$key]);
                      }
                  }

+                 // Retrieve list of primary ports for current protocol
                  $primaryPorts = $processes[0];

+                 /*
+                     Prevent individual listening ports / sockets from overloading using this
```

```
                                shuffle($ports);
                                $ports =
array_values($ports);
                                krsort($ports);
   */
        protected function _createDirectories() {
                $directories = array(

                        '/scripts/cache',
-                       '/scripts/errors'


                );

                foreach ($directories as $directory) {
        }

   /**
-   * Reconfigure process
    *
    * @param string $protocol Proxy protocol (http or
socks)
```

```
simple, persistent internal load balancing
+                                       method for DNAT
ports only (without changing the destination IP). This
also prevents malicious processes from hijacking
+                                       shared ports (when
the combination of SO_REUSEPORT and SO_REUSEADDR are used)
while still allowing a large number of sockets
+                               */
                                shuffle($ports);
                                $ports =
array_values($ports);
                                krsort($ports);
   */
        protected function _createDirectories() {
                $directories = array(
+                       '/scripts',
                        '/scripts/cache',
+                       '/scripts/errors',
+                       '/scripts/iptables',
+                       '/scripts/pid'
                );

                foreach ($directories as $directory) {
        }

   /**
+   * Get process IDs from process name
+   *
+   * @param string $processName Process name
+   *
+   * @return array $processIds Process IDs
+   */
+       protected function _getProcessIds($processName) {
+               $processIds = array();
+               exec('ps -fC ' . $processName . ' 2>&1',
$processes);
+
+               if (!empty($processes[0])) {
+                       unset($processes[0]);
+
+                       foreach ($processes as $process) {
+                               $processColumns =
array_values(array_filter(explode(' ', $process)));
+
+                               if
(!empty($processColumns[1])) {
+                                       $processIds[] =
$processColumns[1];
+                               }
+                       }
+               }
+
+               return $processIds;
+       }
+
+  /**
+   * Reconfigure specific process
    *
    * @param string $protocol Proxy protocol (http or
socks)
+   * @param string $binary Name of system binary
+   * @param string $command Software-specific command for
reconfiguration
```

```
   * @param string $processName Process name
   * @param string $processId Full path to process ID
-  * @param string $aclFile Full path to ACL filename
-  * @param string $aclData ACL data for $aclFile
   * @param integer $delayStart Delay at beginning of reconfiguration
   * @param integer $delayEnd Delay at end of reconfiguration
   *
   * @return
   */
-       protected function _reconfigure($protocol, $processName, $processId, $aclFile, $aclData, $delayStart = 0, $delayEnd = 0) {
                if ($delayStart) {
                        sleep($delayStart);
                }

-               // Reconfigure processes here
```

```
+  * @param string $errorString Reconfiguration error string for error detection
   * @param string $processName Process name
   * @param string $processId Full path to process ID
   * @param integer $delayStart Delay at beginning of reconfiguration
   * @param integer $delayEnd Delay at end of reconfiguration
   *
   * @return
   */
+       protected function _reconfigure($protocol, $binary, $command, $errorString, $processName, $processId, $delayStart = 0, $delayEnd = 0) {
                if ($delayStart) {
                        sleep($delayStart);
                }

+               $killProcesses = array();
+               $shellCommands = array(
+                       '#!' . $this->shell
+               );
+
+               if (!empty($command)) {
+                       exec($command . ' 2>&1', $reconfigureResponse);
+                       $reconfigureResponse = array_shift($reconfigureResponse);
+               }
+
+               if (
+                       (
+                               !empty($reconfigureResponse) &&
+                               !empty($errorString) &&
+                               strpos(strtolower($reconfigureResponse), $errorString) !== false
+                       ) ||
+                       empty($command)
+               ) {
+                       $killProcesses = $this->getProcessIds($processName);
+               }
+
+               foreach ($killProcesses as $killProcess) {
+                       $shellCommands[] = 'kill ' . trim($killProcess);
+               }
+
+               // Run shell commands through bash script
+               if (count($shellCommands > 1)) {
+                       unlink('/scripts/' . $protocol . '.sh');
+                       file_put_contents('/scripts/' . $protocol . '.sh', implode("\n", $shellCommands));
+                       shell_exec('chmod +x /scripts/' . $protocol . '.sh');
+                       shell_exec('/scripts/./' . $protocol . '.sh');
+
```

```diff
                if ($delayEnd) {
                        sleep($delayEnd);
                }
        }

  /**
-  * Reconfigure redundant processes
   *
-  * @return boolean $status
   */
        protected function _applyReconfiguration() {
                // Create writable log and cache directories
                // Apply primary Squid config
                file_put_contents('/etc/squid3/squid.conf', $gatewaysData['data']['squid_conf']);

-               // TODO: Apply 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
                        }
```

```diff
+                       // Process recovery
+                       unlink($processId);
+                       sleep(5);
+                       shell_exec('service ' . $binary . ' start');
+               }

                if ($delayEnd) {
                        sleep($delayEnd);
                }
        }

  /**
+  * 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
                // 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
                        }
                }

-               // Retrieve firewall rules
-               $firewallRules = $this->_configureFirewallRules($gatewaysData);

                // 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
                shell_exec('htpasswd -cb /etc/squid3/passwords default default');
                shell_exec('htpasswd -D /etc/squid3/passwords default');

                }

                // Apply redundant firewall rules to begin seamless reconfiguration
-               unlink('/scripts/iptables/iptables-redundant');
-               touch('/scripts/iptables/iptables-redundant');

-               foreach ($rules as $ruleChunk) {
-                       $saveRules = implode("\n", $ruleChunk);
-                       shell_exec('echo "' . $saveRules . '" >> /scripts/iptables/iptables-redundant');
-               }
-
-               shell_exec('iptables-restore < /scripts/iptables/iptables-redundant');
-               sleep(5);
-               shell_exec('iptables-save');
-
-               // Reconfigure existing SOCKS process first once redundant firewall is applied
                if (
                        !empty($gatewaysData['data']['socks']) &&
                        !empty($this->processes['socks'][0])
                ) {
-                       // Check for alive primary SOCKS 5 port
-                       $socksAlive = false;
-
-                       foreach ($this->processes['socks'][0] as $socksPort) {
-                               if (($socksAlive = $this->_checkPort($proxies[0], $socksPort, 'socks')) === true) {
-                                       break;
```

```
                }

+               $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');

                }

                // 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)
```

```
-                               }
-                       }
-
-                       $this->_reconfigure('socks',
'3proxy.cfg', '/usr/local/etc/3proxy/3proxy.pid',
'/usr/local/etc/3proxy/3proxy.cfg', $gatewaysData['data']
['socks'], 10, 20);
-                       /*
-                       sleep(10);

file_put_contents('/usr/local/etc/3proxy/3proxy.cfg',
$gatewaysData['data']['socks']);
-                       $socksProcess =
array_filter(explode("\n", exec('ps -fC 3proxy.cfg')));
-
-                       if (!empty($socksProcess[0])) {
-                               $splitSocksProcess =
array_values(array_filter(explode(' ',
$socksProcess[0])));
-
-                               if (
-
!empty($splitSocksProcess[1]) &&
-
is_numeric($splitSocksProcess[1])
-                               ) {
-                                       $shell =
'#!/bin/bash';
-                                       $shell .= "\n" .
'kill ' . ($socksAlive ? '-SIGUSR1 ' : '') .
trim($splitSocksProcess[1]);
-
unlink('/scripts/socks.sh');
-
file_put_contents('/scripts/socks.sh', $shell);
-                                       shell_exec('chmod
+x /scripts/socks.sh');
-
shell_exec('/scripts/./socks.sh');
-                               }
-
-                               if (!$socksAlive) {
-
unlink('/usr/local/etc/3proxy/3proxy.pid');
-                                       sleep(10);
-
shell_exec('service 3proxy start');
-                               }
-                       }

-                       // Add raw 20 second delay between
SOCKS and HTTP reconfiguration as necessary
-                       sleep(20);*/
```

```
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
+                       );
```

```
                }

-               /*
-               $mainResponse = shell_exec('squid3 -k reconfigure');
-
-               if (strpos(strtolower($mainResponse), ' error:') !== false) {
-                       exec('ps -fC squid3 2>&1', $processes);

-                       if (!empty($processes[0])) {
-                               unset($processes[0]);

-                               foreach ($processes as $process) {
-                                       $processColumns = array_values(array_filter(explode(' ', $process)));

-                                       if (!empty($processColumns[1])) {
-                                               $killProcesses[] = $processColumns[1];
-                                       }
-                               }

-                               if (!empty($killProcesses)) {
-                                       $shell = '#!/bin/bash';
-
-                                       foreach ($killProcesses as $killProcess) {
-                                               $shell .= "\n" . 'kill ' . trim($killProcess);
-                                       }
-
-                                       unlink('/scripts/http.sh');
-                                       file_put_contents('/scripts/http.sh', $shell);
-                                       shell_exec('chmod +x /scripts/http.sh');
-                                       shell_exec('/scripts/./http.sh');
-                               }
```

```
                }

+               // Reconfigure main HTTP instance
+               $this->_reconfigure(
+                       'http',
+                       'squid3',
+                       'squid3 -k reconfigure',
+                       ' error:',
+                       'squid3',
+                       '/var/run/squid3.pid',
+                       0,
+                       70
+               );

+               $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]);
```

```
                        }

-                       unlink('/var/run/squid3.pid');
-                       sleep(3);
-                       shell_exec('squid3 start');
-               }
-
-               sleep(70);
-               unlink('/scripts/iptables/iptables-redundant');
-               shell_exec('php /scripts/firewall_redundant.php ports6+ports7+ports8+ports9+ports10');
-
-               foreach (range(1, 5) as $instance) {
-                       $instance = $instance == 1 ? '' : $instance;
-                       exec('squid3-redundant' . $instance . ' -k reconfigure -f /etc/squid3/squid-redundant' . $instance . '.conf 2>&1', $redundantResponse);
-                       $redundantResponse = array_shift($redundantResponse);
-
-                       if (strpos(strtolower($redundantResponse), ' error:') !== false) {
-                               exec('ps -fC squid3-redundant 2>&1', $processes);
-
-                               if (!empty($processes[0])) {
-                                       unset($processes[0]);
-
-                                       foreach ($processes as $process) {
-                                               $processColumns = array_values(array_filter(explode(' ', $process)));
-
-                                               if (
-                                                       !empty($processColumns[1]) &&
-                                                       in_array('/etc/squid3/squid-redundant' . $instance . '.conf', $processColumns)
-                                               ) {
-                                                       $killProcesses[] = $processColumns[1];
-                                               }
-                                       }
-
-                                       if (!empty($killProcesses)) {
-                                               $shell = '#!/bin/bash';

-                                               foreach ($killProcesses as $killProcess) {
-                                                       $shell .= "\n" . 'kill ' . trim($killProcess);
```

```
                        }

+                       // Get list of active redundant ports for firewall configuration
+                       $activeRedundantPorts = array();

+                       foreach ($redundantProcesses as $key => $redundantProcess) {
+                               if (in_array($key, $activeRedundantProcesses)) {
```

```
-                                               }
-
-                                               unlink('/scripts/http.sh');
-                                               file_put_contents('/scripts/http.sh', $shell);
-                                               shell_exec('chmod +x /scripts/http.sh');
-                                               shell_exec('/scripts/./http.sh');
-                                       }
                                }
-
-                               shell_exec('rm /var/run/squid-redundant' . $instance . '.pid');
-                               sleep(3);
-                               shell_exec('squid3-redundant' . $instance . ' start -f /etc/squid3/squid-redundant' . $instance . '.conf');
                        }
-               }
-
-               sleep(70);
-               shell_exec('php /scripts/firewall_redundant.php ports1+ports2+ports3+ports4+ports5');
-
-               foreach (range(6, 10) as $instance) {
-                       exec('squid3-redundant' . $instance . ' -k reconfigure -f /etc/squid3/squid-redundant' . $instance . '.conf 2>&1', $redundantResponse);
-                       $redundantResponse = array_shift($redundantResponse);
-
-                       if (strpos(strtolower($redundantResponse), ' error:') !== false) {
-                               exec('ps -fC squid3-redundant 2>&1', $processes);

-                               if (!empty($processes[0])) {
-                                       unset($processes[0]);
-
-                                       foreach ($processes as $process) {
-                                               $processColumns = array_values(array_filter(explode(' ', $process)));
-
-                                               if (
-                                                       !empty($processColumns[1]) &&
-                                                       in_array('/etc/squid3/squid-redundant' . $instance . '.conf', $processColumns)
-                                               ) {
-                                                       $killProcesses[] = $processColumns[1];
```

```
+                                       $activeRedundantPorts = array_merge($activeRedundantPorts, $redundantProcesses[$key]);
                                }
                        }

+                       $overridePorts = array(
+                               'http' => array_merge(array(
+                                       '80',
+                                       '8888',
+                                       '55555'
+                               ), $activeRedundantPorts),
+                               'socks' => array(
+                                       '1090'
+                               )
+                       );
+
-                        }
-                    }
-
-                    if (!empty($killProcesses)) {
-                        $shell = '#!/bin/bash';
-
-                        foreach ($killProcesses as $killProcess) {
-                            $shell .= "\n" . 'kill ' . trim($killProcess);
-                        }
-
-                        unlink('/scripts/http.sh');
-                        file_put_contents('/scripts/http.sh', $shell);
-                        shell_exec('chmod +x /scripts/http.sh');
-                        shell_exec('/scripts/./http.sh');
-                    }
-                }
-
-                shell_exec('rm /var/run/squid-redundant' . $instance . '.pid');
-                sleep(3);
-                shell_exec('squid3-redundant' . $instance . ' start -f /etc/squid3/squid-redundant' . $instance . '.conf');
             }
         }

-        sleep(70);
-        shell_exec('php /scripts/firewall.php override');

         if (!empty($gatewaysData['data']['socks-redundant'])) {
-            sleep(15);
```

```
+            $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']);
```

```
-                        shell_exec('php /scripts/firewall.php');
-                        exec('curl --socks5-hostname ' . $proxies[0] . ':1090' . ' http://socks/ -v --connect-timeout 4 --max-time 4 2>&1', $socksResponseRedundant);
-                        $socksResponseRedundant = end($socksResponseRedundant);
-                        $socksAliveRedundant = (strpos(strtolower($socksResponseRedundant), 'empty reply ') !== false);
-                        file_put_contents('/usr/local/etc/3proxy/3proxy-redundant.cfg', $gatewaysData['data']['socks-redundant']);
-                        $socksProcessRedundant = array_filter(explode("\n", exec('ps -fC 3proxy-redundant.cfg')));
-
-                        if (!empty($socksProcessRedundant[0])) {
-                                $splitSocksProcessRedundant = array_values(array_filter(explode(' ', $socksProcessRedundant[0])));
-
-                                if (
-                                        !empty($splitSocksProcessRedundant[1]) &&
-                                        is_numeric($splitSocksProcessRedundant[1])
-                                ) {
-                                        $shell = '#!/bin/bash';
-                                        $shell .= "\n" . 'kill ' . ($socksAliveRedundant ? '-SIGUSR1 ' : '') . trim($splitSocksProcessRedundant[1]);
-                                        unlink('/scripts/socks.sh');
-                                        file_put_contents('/scripts/socks.sh', $shell);
-                                        shell_exec('chmod +x /scripts/socks.sh');
-                                        shell_exec('/scripts/./socks.sh');
-                                }

-                                if (!$socksAliveRedundant) {
-                                        unlink('/usr/local/etc/3proxy/3proxy-redundant.pid');
-                                        sleep(10);
-                                        shell_exec('service 3proxy-redundant start');
-                                }
-                        }
                }

-               shell_exec('php /scripts/firewall.php
```

```

+                       // 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,
  override');

                unlink('/scripts/pid/reconfigure.pid');
-               die();
-               */

        }

  /**
        }

  }
-
- // Debug code, remove later
- $api = 'https://ghostproxies.com/gateways/api.json';
- $processes = array(
-       'http' => array(
-               array(
-                       '80',
-                       '8888',
-                       '55555'),
-               array(
-                       '4444',
-                       '7777'
-               ),
-               array(
-                       '55540',
-                       '55541'
-               ),
-               array(
-                       '55546',
-                       '55547'
-               ),
-               array(
-                       '55520',
-                       '55521'
-               ),
-               array(
-                       '55524',
-                       '55525'
-               ),
-               array(
-                       '55526',
-                       '55527'
-               ),
-               array(
-                       '55528',
-                       '55529'
-               ),
-               array(
-                       '55530',
-                       '55531'
-               ),
-               array(
-                       '55532',
-                       '55533'
-               ),
-               array(
-                       '55534',
-                       '55535'
```

```
  'elastic');
+
+               // Remove reconfiguration process ID
                unlink('/scripts/pid/reconfigure.pid');
+
+               // Memory cleanup
+               gc_collect_cycles();
+
+               return true;
        }

  /**
        }

  }
```

```
-                       )
-               ),
-               'socks' => array(
-                       array(
-                               '1085'
-                       ),
-                       array(
-                               '1090'
-                       )
-               )
- );
- $sshPorts = array(
-       '22',
-       '5000'
- );
-
- $dynamicProxyReconfiguration = new DynamicProxyReconfiguration($api, $processes, $sshPorts);
- $status = $dynamicProxyReconfiguration->start('apply_reconfiguration');
-
- print_r($status);
- die();
  ?>                                                            ?>
-
- TODO:
- Add support for additional HTTP and SOCKS proxy servers
- Apply proxy redundancy configuration from API to all redundant processes
- Start additional redundancy processes when CPU exceeds x%, provide option to terminate redundant processes if average CPU usage is below x%
```

0 comments on commit d09e8c2