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

Update configuration for all redundant processes

master

Browse files

parsonsbots committed on Apr 14

1 parent 05bbaa9 commit 4bc9eb21afbab7349f86c6716986a19c7666fc9d

Showing **1 changed file** with **49 additions** and **44 deletions**.

Unified Split

93 dynamic_reconfiguration.php

```
             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');
     }


         }

         $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']) ||
```

```
             return false;
         }

+        $this->gatewaysData = json_decode(file_get_contents('/scripts/cache/gatewaysData'), true);

+        if (empty($this->gatewaysData['data']['proxies'])) {
             return false;
         }

         $overridePorts = array(
             'http' => $this->processes['http']
         );
+        $firewallRules = $this->_configureFirewallRules(false, $overridePorts);
         $this->_applyFirewallRules($firewallRules, 'elastic');
     }


         }

         $gatewaysJsonData = shell_exec("curl " . $this->api . " --connect-timeout 30");
+        $this->gatewaysData = json_decode($gatewaysJsonData, true);

         // Log API error timestamp
         if (
+            empty($this->gatewaysData['data']) ||
             !is_dir('/etc/squid3')
         ) {
+                file_put_contents('/scripts/errors/api-error-' . time(), $this->gatewaysData);
                 unlink('/scripts/pid/reconfigure.pid');
             return false;
         }

         // Require Squid and sysctl configurations from API
         if (
+            empty($this->gatewaysData['data']['squid_conf']) ||
-                       empty($gatewaysData['data']['squid_redundant_conf']) ||
-                       empty($gatewaysData['data']['sysctl_conf'])
                ) {
                        unlink('/scripts/pid/reconfigure.pid');
                        return false;
                // 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');
                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');
-
```

```
+                       empty($this->gatewaysData['data']['sysctl_conf'])
                ) {
                        unlink('/scripts/pid/reconfigure.pid');
                        return false;
                // Create new reconfiguration process ID
                file_put_contents('/scripts/pid/reconfigure.pid', time());

                // Save and apply sysctl settings
+               file_put_contents('/etc/sysctl.conf', $this->gatewaysData['data']['sysctl_conf']);
                shell_exec('sysctl -p');

                // Don't run reconfiguration if there aren't any ACLs to apply to proxy IPs
+               $proxies = $this->gatewaysData['data']['proxies'];

                if (empty($proxies[0])) {
                        unlink('/scripts/pid/reconfigure.pid');
                shell_exec('rm -rf /etc/squid3/users/');
                shell_exec('mkdir -m 777 /etc/squid3/users/');

+               if (!empty($this->gatewaysData['data']['files'])) {
+                       foreach ($this->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(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);
                        }
                }

                // 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',
                                'service 3proxy start',
                                '3proxy.cfg',
                                '/usr/local/etc/3proxy/3proxy.pid',
                                5,
-                               15
                        );
                }

                        'squid3',
                        '/var/run/squid3.pid',
                        0,
-                       65
                );

                $redundantProcesses = $this-
```

```
file_put_contents('/etc/squid3/proxy_ip_acl.conf', implode("\n", $this->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($this->gatewaysData['data']['users'])) {
+                       foreach ($this->gatewaysData['data']['users'] as $username => $password) {
                                shell_exec('htpasswd -b /etc/squid3/passwords ' . $username . ' ' . $password);
                        }
                }

                // Reconfigure existing SOCKS processes first once redundant firewall is applied
                if (
+                       !empty($this->gatewaysData['data']['socks']) &&
                        !empty($this->processes['socks'][0])
                ) {
                        // Reconfigure main SOCKS instance
                        $this->_reconfigure(
                                'socks',
                                'service 3proxy start',
                                '3proxy.cfg',
                                '/usr/local/etc/3proxy/3proxy.pid',
                                5,
+                               15,
+                               '/usr/local/etc/3proxy/3proxy.cfg',
+                               $this->gatewaysData['data']['socks']
                        );
                }

                        'squid3',
                        '/var/run/squid3.pid',
                        0,
+                       65,
+                       '/etc/squid3/squid.conf',
+                       str_replace('[pid]', 'pid_filename /var/run/squid3.pid', str_replace('[ports]', 'http_port ' . implode("\n" . 'http_port ', $this->processes['http'][0]), $this->gatewaysData['data']['squid_conf']))
                );

                $redundantProcesses = $this-
```

```
>processes['http']; // TODO: chunk and reconfigure HTTP and SOCKS processes simultaneously
                                )
                        );

-                       $firewallRules = $this->_configureFirewallRules($gatewaysData, false, $overridePorts);
                        $this->_applyFirewallRules($firewallRules, 'elastic');

                        foreach ($redundantProcessRange as $redundantProcess) {
                                // Reconfigure redundant HTTP instances
                                $this->_reconfigure(
                                        'http',
-                                       'squid3-redundant' . $redundantProcess . ' start -f /etc/squid3/squid-redundant' . $redundantProcess . '.conf',
                                        'squid3-redundant' . $redundantProcess,
                                        '/var/run/squid-redundant' . $redundantProcess . '.pid',
                                        0,
-                                       0
                                );
                        }

                $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',
                                'service 3proxy-redundant start',
                                '3proxy-redundant.cfg',
                                '/usr/local/etc/3proxy/3proxy-redundant.pid',
```

```
>processes['http']; // TODO: chunk and reconfigure HTTP and SOCKS processes simultaneously
                                )
                        );

+                       $firewallRules = $this->_configureFirewallRules(false, $overridePorts);
                        $this->_applyFirewallRules($firewallRules, 'elastic');

                        foreach ($redundantProcessRange as $redundantProcess) {
                                // Reconfigure redundant HTTP instances
                                $this->_reconfigure(
                                        'http',
+                                       'squid3-redundant' . $redundantProcess . ' start -f ' . $configurationFile,
                                        'squid3-redundant' . $redundantProcess,
                                        '/var/run/squid-redundant' . $redundantProcess . '.pid',
                                        0,
+                                       0,
+                                       '/etc/squid3/squid-redundant' . $redundantProcess . '.conf',
+                                       str_replace('[pid]', 'pid_filename /var/run/squid-redundant' . $redundantProcess . '.pid', str_replace('[ports]', 'http_port ' . implode("\n" . 'http_port ', $this->processes['http'][$redundantProcess]), $this->gatewaysData['data']['squid_conf']))
                                );
                        }

                $overridePorts = array(
                        'http' => $this->processes['http']
                );
+               $firewallRules = $this->_configureFirewallRules(false, $overridePorts);
                $this->_applyFirewallRules($firewallRules, 'elastic');

+               if (!empty($this->gatewaysData['data']['socks-redundant'])) {
                        // Reconfigure redundant SOCKS instance
                        $this->_reconfigure(
                                'socks',
                                'service 3proxy-redundant start',
                                '3proxy-redundant.cfg',
                                '/usr/local/etc/3proxy/3proxy-redundant.pid',
                                10,
-                               0
                        );
                }

  /**
   * Configure firewall rules
   *
-  * @param array $gatewaysData Data from gateway API response
   * @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',

                // Apply custom firewall rules from API (e.g. disabling ports for specific destinations)
                if (
-                       !empty($gatewaysData['data']['firewall_filter']) &&
-                       is_array($gatewaysData['data']['firewall_filter'])
                ) {
-                       foreach ($gatewaysData['data']['firewall_filter'] as $rule) {
                                $rules[] = $rule;
                        }
                }

                                // 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]);
                                        }
                                }
   * @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
```

```
                                10,
+                               0,
+                               '/usr/local/etc/3proxy/3proxy-redundant.cfg',
+                               $this->gatewaysData['data']['socks-redundant']
                        );
                }

  /**
   * Configure firewall rules
   *
   * @param boolean $redundant Route to redundant process ports only if true, all ports if false
   * @param array $ports Override default base ports
   *
   * @return array $rules Firewall rules
   */
+       protected function _configureFirewallRules($redundant = false, $overridePorts = array()) {
                // Begin input filter rules
                $rules = array(
                        '*filter',

                // Apply custom firewall rules from API (e.g. disabling ports for specific destinations)
                if (
+                       !empty($this->gatewaysData['data']['firewall_filter']) &&
+                       is_array($this->gatewaysData['data']['firewall_filter'])
                ) {
+                       foreach ($this->gatewaysData['data']['firewall_filter'] as $rule) {
                                $rules[] = $rule;
                        }
                }

                                // Remove ports from routing list that are refusing connections
                                foreach ($ports as $key => $port) {
+                                       if ($this->_checkPort($this->gatewaysData['data']['proxies'][0], $port, $protocol) === false) {
                                                unset($ports[$key]);
                                        }
                                }
   * @param string $processId Full path to process ID
   * @param integer $delayStart Delay at beginning of reconfiguration
   * @param integer $delayEnd Delay at end of reconfiguration
+  * @param string $configurationFile Full path to process configuration file
+  * @param string $configurationData Process configuration data
   *
   * @return
```

```
	*/
-	protected function _reconfigure($protocol, $startCommand, $processName, $processId, $delayStart = 0, $delayEnd = 0) {
		if ($delayStart) {
			sleep($delayStart);
		}
			shell_exec('/scripts/./' . $protocol . '.sh');
		}

		// Process recovery
		unlink($processId);
		sleep(1);
```

```
	*/
+	protected function _reconfigure($protocol, $startCommand, $processName, $processId, $delayStart = 0, $delayEnd = 0, $configurationFile, $configurationData) {
		if ($delayStart) {
			sleep($delayStart);
		}
			shell_exec('/scripts/./' . $protocol . '.sh');
		}

+		// Update configuration file for process
+		if (
+			!empty($configurationFile) &&
+			!empty($configurationData)
+		) {
+			file_put_contents($configurationFile, $configurationData);
+		}
+
		// Process recovery
		unlink($processId);
		sleep(1);
```

0 comments on commit `4bc9eb2`