




Updates

 master

 parsonsbots committed 26 seconds ago


1 parent [2bd4388](#) commit [37e75ec92a3a600e16c22329c121b9039a9ee6c3](#)

[Browse files](#)

 Showing **2 changed files** with **514 additions** and **534 deletions**.

Unified

Split

▼ 9 README.md 

42

ini_set('max_execution_time', 595); // Set time limit to 10 minutes for reconfiguration

43

require_once('dynamic_reconfiguration.php');

44

- \$api =

45

'https://ghostproxies.com/gateways/api.json';

46

\$processes = array(

47

'http' => array(

48

array('80', '8888', '55555'), //

49

)

50

);

51

);

52

);

53

);

54

);

55

);

56

);

57

);

58

);

59

);

60

);

61

);

62

);

63

);

64

);

65

);

66

);

67

);

68

);

69

);

70

);

71

);

72

);

73

);

74

);

75

);

76

);

77

);

78

);

79

);

80

);

81

);

82

);

83

);

84

);

85

);

86

);

87

);

88

);

89

);

90

);

91

);

92

);

93

);

94

);

95

);

96

);

97

);

98

);

99

);

100

);

101

);

102

);

103

);

104

);

105

);

106

);

107

);

108

);

109

);

110

);

111

);

112

);

113

);

114

);

115

);

116

);

117

);

118

);

119

);

120

);

121

);

122

);

123

);

124

);

125

);

126

);

127

);

128

);

129

);

130

);

131

);

132

);

133

);

134

);

135

);

136

);

137

);

138

);

139

);

140

);

141

);

142

);

143

);

144

);

145

);

146

);

147

);

148

);

149

);

150

);

151

);

152

);

153

);

154

);

155

);

156

);

157

);

158

);

159

);

160

);

161

);

162

);

163

);

164

);

165

);

166

);

167

);

168

);

169

);

170

);

171

);

172

);

173

);

174

);

175

);

176

);

177

);

178

);

179

);

180

);

181

);

182

);

183

);

184

);

185

);

186

);

187

);

188

);

189

);

190

);

191

);

192

);

193

);

194

);

195

);

196

);

197

);

198

);

199

);

200

);

201

);

202

);

203

);

204

);

205

);

206

);

207

);

208

);

209

);

210

);

211

);

212

);

213

);

214

);

215

);

216

);

217

);

218

);

219

);

220

);

221

);

222

);

223

);

224

);

225

);

226

);

227

);

228

);

229

);

230

);

231

);

232

);

233

);

234

);

235

);

236

);

237

);

238

);

239

);

240

);

241

);

242

);

243

);

244

);

245

);

246

);

247

);

248

);

249

);

250

);

251

);

252

);

253

);

254

);

255

);

256

);

257

);

258

);

259

);

260

);

261

);

262

);

263

);

264

);

265

);

266

);

267

);

268

);

269

);

270

);

271

);

272

);

273

);

274

);

275

);

276

);

277

);

278

);

279

);

280

);

281

);

282

);

283

);

284

);

285

);

286

);

287

);

288

);

289

);

290

);

291

);

292

);

293

);

294

);

295

);

296

);

297

);

298

);

299

);

300

);

301

);

302

);

303

);

304

);

305

);

306

);

307

);

308

);

309

);

310

);

311

);

312

);

313

);

314

);

315

);

316

);

317

);

318

);

319

);

320

);

321

);

322

);

323

);

324

);

325

);

326

);

327

);

328

);

329

);

330

);

331

);

332

);

333

);

334

);

335

);

336

);

337

);

338

);

339

);

340

);

341

);

342

);

343

);

344

);

345

);

346

);

347

);

348

);

349

);

350

);

351

);

352

);

353

);

354

);

355

);

356

);

357

);

358

);

359

);

360

);

361

);

362

);

363

);

364

);

365

);

366

);

367

);

368

);

369

);

370

);

371

);

372

);

373

);

374

);

375

);

376

);

377

);

378

);

379

);

380

);

381

);

382

);

383

);

384

);

385

);

386

);

387

);

388

);

389

);

390

);

391

);

392

);

393

);

394

);

395

);

396

);

397

);

398

);

399

);

400

);

401

);

402

);

403

);

404

);

405

);

406

);

407

);

408

);

409

);

410

);

411

);

412

);

413

);

414

);

415

);

416

);

417

);

418

);

419

);

420

);

421

);

422

);

423

);

424

);

425

);

426

);

427

);

428

);

429

);

430

);

431

);

432

);

433

);

434

);

435

);

436

);

437

);

438

);

439

);

440

);

441

);

442

);

443

);

444

);

445

);

446

);

447

);

448

);

449

);

450

);

451

);

452

);

453

);

454

);

455

);

456

);

457

);

458

);

459

);

460

);

461

);

462

);

463

);

464

);

465

);

466

);

467

);

468

);

469

);

470

);

471

);

472

);

473

);

474

);

475

);

476

);

477

);

478

);

479

);

480

);

481

);

482

);

483

);

484

);

485

);

486

);

487

);

488

);

489

);

490

);

491

);

492

);

493

);

494

);

495

);

496

);

497

);

498

);

499

);

500

);

501

);

502

);

503

);

504

);

505

);

506

);

507

);

508

);

509

);

510

);

511

);

512

);

513

);

514

);

515

);

516

);

517

);

518

);

519

);

520

);

521

);

522

);

523

);

524

);

525

);

526

);

527

);

528

);

529

);

530

);

531

);

532

);

533

);

534

);

535

);

536

);

537

);

538

);

539

);

540

);

541

);

542

);

543

);

544

);

545

);

546

);

547

);

548

);

549

);

550

);

551

);

552

);

553

);

554

);

555

);

556

);

557

);

558

);

559

);

560

);

561

);

562

);

563

);

564

);

565

);

566

);

567

);

568

);

569

);

570

);

571

);

572

);

573

);

574

);

575

);

576

);

577

);

578

);

579

);

580

);

581

);

582

);

583

);

584

);

585

);

586

);

587

);

588

);

589

);

590

);

591

);

592

);

593

);

594

);

595

);

596

);

597

);

598

);

599

);

600

);

601

);

602

);

603

);

604

);

605

);

606

);

607

);

608

);

609

);

610

);

611

);

612

);

613

);

614

);

615

);

616

);

617

);

618

);

619

);

620

);

621

);

622

);

623

);

624

);

625

);

626

);

627

);

628

);

629

);

630

);

631

);

632

);

633

);

634

);

635

);

636

);

637

);

638

);

639

);

640

);

641

);

642

);

643

);

644

);

645

);

646

);

647

);

648

);

649

);

650

);

651

);

652

);

653

);

654

);

655

);

656

);

657

);

658

);

659

);

660

);

661

);

662

);

663

);

664

);

665

);

666

);

667

);

668

);

669

);

670

);

671

);

672

);

673

);

674

);

675

);

676

);

677

);

678

);

679

);

680

);

681

);

682

);

683

);

684

);

685

);

686

);

687

);

688

);

689

);

690

);

691

);

692

);

693

);

694

);

695

);

696

);

697

);

698

);

699

);

700

);

701

);

702

);

703

);

704

);

705

);

706

);

707

);

708

);

709

);

710

);

711

);

712

);

713

);

714

);

715

);

716

);

717

);

718

);

719

);

720

);

721

);

722

);

723

);

724

);

725

);

726

);

727

);

728

);

729

);

730

);

731

);

732

);

733

);

734

);

735

);

736

);

737

);

738

);

739

);

740

);

741

);

742

);

743

);

744

);

745

);

746

);

747

);

748

);

749

);

750

);

751

);

752

);

753

);

754

);

755

);

756

);

757

);

758

);

759

);

760

);

761

);

762

);

763

);

764

);

765

);

766

);

767

);

768

);

769

);

770

);

771

);

772

);

773

);

774

);

775

);

776

);

777

);

778

);

779

);

780

);

781

);

782

);

783

);

784

);

785

);

786

);

787

);

788

);

789

);

790

);

791

);

792

);

793

);

794

);

795

);

796

);

797

);

798

);

799

);

800

);

801

);

802

);

803

);

804

);

805

);

806

);

807

);

808

);

809

);

810

);

811

);

812

);

813

);

814

);

815

);

816

);

817

);

818

);

819

);

820

);

821

);

822

);

823

);

824

);

825

);

826

);

827

);

828

);

829

);

830

);

831

);

832

);

833

);

834

);

835

);

836

);

837

);

838

);

839

);

840

);

841

);

842

);

843

);

844

);

845

);

846

);

847

);

848

);

849

);

850

);

851

);

852

);

853

);

854

);

855

);

856

);

857

);

858

);

859

);

860

);

861

);

862

);

863

);

864

);

865

);

866

);

867

);

868

);

869

);

870

);

871

);

872

);

873

);

874

);

875

);

876

);

877

);

878

);

879

);

880

);

881

);

882

);

883

);

884

);

885

);

886

);

887

);

```

10 - * @copyright 2019 Will Parsons
11 - * @license https://github.com/parsonsbots/dynamic-
    proxy-node-reconfiguration/blob/master/LICENSE MIT License
12 - * @link https://parsonsbots.com
13 - * @link https://eightomic.com
14 - */
15 - class DynamicProxyReconfiguration {
16 -
17 -     public $api;
18 -     public $processes;
19 -     public $shell;
20 -     public $sshPorts;
21 -
22 -     public function __construct($api, $processes,
    $shell, $sshPorts) {
23 -         $this->api = $api;
24 -         $this->processes = $processes;
25 -         $this->shell = $shell;
26 -         $this->sshPorts = $sshPorts;
27 -     }
28 -
29 - /**
30 -  * Apply firewall
31 -  *
32 -  * @return boolean
33 -  */
34 -     protected function _applyFirewall() {
35 -         if
    (file_exists('/scripts/pid/reconfigure.pid')) {
36 -             return false;
37 -         }
38 -
39 -         $this->gatewaysData =
    json_decode(file_get_contents('/scripts/cache/gatewaysData
    '), true);
40 -
41 -         if (empty($this->gatewaysData['data']
    ['proxy_ips'])) {
42 -             return false;
43 -         }
44 -
45 -         $overridePorts = array(
46 -             'http' => $this->processes['http']
47 -         );
48 -         $firewallRules = $this-
    >_configureFirewallRules(false, $overridePorts);
49 -         $this->_applyFirewallRules($firewallRules,
    'elastic');
50 -     }
51 -
52 - /**
53 -  * Apply firewall rules

```

```

10 +         $this->api = $api;
11 +         $this->processes = $processes;
12 +
13 +         $this->shell = $shell;
14 +         $this->sshPorts = $sshPorts;
15 +
16 +         /**
17 +         * Apply firewall
18 +         *
19 +         * @return boolean
20 +         */
21 +         protected function _applyFirewall() {
22 +             if
    (file_exists('/scripts/pid/reconfigure.pid')) {
23 +                 return false;
24 +             }
25 +
26 +             $this->gatewaysData =
    json_decode(file_get_contents('/scripts/cache/gatewaysData
    '), true);
27 +
28 +             if (empty($this-
    >gatewaysData['data']['proxy_ips'])) {
29 +                 return false;
30 +             }
31 +
32 +             $overridePorts = array(
33 +                 'http' => $this-

```

```

54 - *
55 - * @param array $firewallRules Firewall rules
56 - * @param string $ruleSet Firewall rule set
57 - *
58 - * @return
59 - */
60 -     protected function
61 _applyFirewallRules($firewallRules, $ruleSet) {
62 -         unlink('/scripts/iptables/iptables-' .
63 $ruleSet);
64 -         touch('/scripts/iptables/iptables-' .
65 $ruleSet);
66 -         foreach ($firewallRules as $ruleChunk) {
67             $saveRules = implode("\n",
68 $ruleChunk);
69 -             shell_exec('echo "' . $saveRules .
70 '" >> /scripts/iptables/iptables-' . $ruleSet);
71 -         }
72
73 - /**
74 - * Apply seamless processes reconfiguration
75 - *
76 - * @return boolean
77 - */
78 -     protected function _applyReconfiguration() {
79 -         $this->_createDirectories();

```

```

>processes['http']
34 +         );
35 +         $firewallRules = $this->
36 _configureFirewallRules(false, $overridePorts);
37 +         $this->
38 _applyFirewallRules($firewallRules, 'elastic');
39 +
40 +     /**
41 +      * Apply firewall rules
42 +      *
43 +      * @param array $firewallRules Firewall rules
44 +      * @param string $ruleSet Firewall rule set
45 +      *
46 +      * @return
47 +      */
48 +     protected function
49 _applyFirewallRules($firewallRules, $ruleSet) {
50 +
51 +         unlink('/scripts/iptables/iptables-' . $ruleSet);
52 +         touch('/scripts/iptables/iptables-' .
53 $ruleSet);
54 +
55 +         foreach ($firewallRules as
56 $ruleChunk) {
57 +             $saveRules = implode("\n",
58 $ruleChunk);
59 +             shell_exec('echo "' .
60 $saveRules . '" >> /scripts/iptables/iptables-' .
61 $ruleSet);
62 +         }
63 +
64 +         shell_exec('iptables-restore <
65 /scripts/iptables/iptables-' . $ruleSet);
66 +         return;
67 +     }
68 +
69 +     /**
70 +      * Apply seamless processes reconfiguration
71 +      *
72 +      * @return boolean
73 +      */
74 +     protected function _applyReconfiguration()
75 {
76 +         $this->_createDirectories();
77 +
78 +         if

```

```

80
81 -         if
82   (file_exists('/scripts/pid/reconfigure.pid')) {
83
84 -             $lastRan =
85   file_get_contents('/scripts/pid/reconfigure.pid');
86
87 -             if ($lastRan < strtotime('-15
88   minutes')) {
89
90
91   unlink('/scripts/pid/reconfigure.pid');
92 -             } else {
93
94   return false;
95
96   }
97
98 -             $gatewaysJsonData = shell_exec("curl " .
99   $this->api . " --connect-timeout 30");
100
101 -             $this->gatewaysData =
102   json_decode($gatewaysJsonData, true);
103
104 -             if (
105
106   empty($this->gatewaysData['data'])
107
108   ||
109
110   !is_dir('/etc/squid3')
111
112   ) {
113
114   file_put_contents('/scripts/errors/api-error-' . time(),
115   $this->gatewaysData);
116
117   unlink('/scripts/pid/reconfigure.pid');
118
119   return false;
120
121   }
122
123 -             file_put_contents('/scripts/cache/gatewaysData',
124   $gatewaysJsonData);
125
126 -             file_put_contents('/scripts/pid/reconfigure.pid', time());
127
128 -             if (!empty($this->gatewaysData['data']
129
130   (file_exists('/scripts/pid/reconfigure.pid')) {
131
132   +             $lastRan =
133   file_get_contents('/scripts/pid/reconfigure.pid');
134
135   +             if ($lastRan <
136   strtotime('-15 minutes')) {
137
138   +             unlink('/scripts/pid/reconfigure.pid');
139   +             } else {
140
141   +                 return false;
142   +             }
143
144   +             $gatewaysJsonData =
145   shell_exec("curl " . $this->api . " --connect-timeout
146   30");
147
148   +             $this->gatewaysData =
149   json_decode($gatewaysJsonData, true);
150
151   +             if (
152
153   +                 empty($this->
154   >gatewaysData['data']) ||
155   +                 !is_dir('/etc/squid3')
156   +                 ) {
157
158   +                 file_put_contents('/scripts/errors/api-error-' . time(),
159   $this->gatewaysData);
160
161   +                 unlink('/scripts/pid/reconfigure.pid');
162
163   +                 return false;
164   +             }
165
166   +             file_put_contents('/scripts/cache/gatewaysData',
167   $gatewaysJsonData);
168
169   +             file_put_contents('/scripts/pid/reconfigure.pid', time());
170
171   +             if (!empty($this->
172   >gatewaysData['data']['server'])) {
173
174   +                 file_put_contents('/etc/sysctl.conf', implode("\n", $this->
175   >gatewaysData['data']['server']));
176   +                 shell_exec('sysctl -p');
177   +             }
178
179   +             $proxyIps = $this->
180   >gatewaysData['data']['proxy_ips'];
181
182   +             if (empty($proxyIps[0])) {

```

```

107 - ['server'])) {
108 -     file_put_contents('/etc/sysctl.conf', implode("\n", $this->gatewaysData['data']['server']));
109 -     shell_exec('sysctl -p');
110 - }
111 - $proxyIps = $this->gatewaysData['data']['proxy_ips'];
112 -
113 - if (empty($proxyIps[0])) {
114 -     unlink('/scripts/pid/reconfigure.pid');
115 -     return false;
116 - }
117 -
118 - shell_exec('rm -rf /etc/squid3/users/');
119 - shell_exec('mkdir -m 777 /etc/squid3/users/');
120 -
121 - if (!empty($this->gatewaysData['data']['http']['files'])) {
122 -     foreach ($this->gatewaysData['data']['http']['files'] as $file) {
123 -         shell_exec('mkdir -m 777 ' . str_replace(array('s.txt', 'd.txt'), '', $file['path']));
124 -         shell_exec('touch ' . $file['path']);
125 -         file_put_contents($file['path'], $file['contents']);
126 -     }
127 - }
128 -
129 - $firewallRules = $this->_configureFirewallRules(true);
130 - shell_exec('rm /etc/squid3/proxy_ip_acl.conf');
131 - shell_exec('touch /etc/squid3/proxy_ip_acl.conf');
132 - file_put_contents('/etc/squid3/proxy_ip_acl.conf', $this->

```

```

101 +     unlink('/scripts/pid/reconfigure.pid');
102 +     return false;
103 + }
104 +
105 +     shell_exec('rm -rf /etc/squid3/users/');
106 +     shell_exec('mkdir -m 777 /etc/squid3/users/');
107 +
108 +     if (!empty($this->gatewaysData['data']['http']['files'])) {
109 +         foreach ($this->gatewaysData['data']['http']['files'] as $file) {
110 +             shell_exec('mkdir -m 777 ' . str_replace(array('s.txt', 'd.txt'), '', $file['path']));
111 +             shell_exec('touch ' . $file['path']);
112 +             file_put_contents($file['path'], $file['contents']);
113 +         }
114 +     }
115 +
116 +     $firewallRules = $this->_configureFirewallRules(true);
117 +     shell_exec('rm /etc/squid3/proxy_ip_acl.conf');
118 +     shell_exec('touch /etc/squid3/proxy_ip_acl.conf');
119 +     file_put_contents('/etc/squid3/proxy_ip_acl.conf', $this->gatewaysData['data']['http']['acls']);
120 +     shell_exec('htpasswd -cb /etc/squid3/passwords default default');
121 +     shell_exec('htpasswd -D /etc/squid3/passwords default');
122 +
123 +     if (!empty($this->gatewaysData['data']['http']['users'])) {
124 +         foreach ($this->gatewaysData['data']['http']['users'] as $username => $password) {
125 +             shell_exec('htpasswd -b /etc/squid3/passwords ' . $username . ' ' . $password);
126 +         }
127 +     }
128 +
129 +     $this->_applyFirewallRules($firewallRules, 'redundant');

```

```

133 >gatewaysData['data']['http']['acls']);
134 -         shell_exec('htpasswd -cb
/etc/squid3/passwords default default');
135
136 -         if (!empty($this->gatewaysData['data']
['http']['users'])) {
137 -             foreach ($this->
>gatewaysData['data']['http']['users'] as $username =>
$password) {
138 -                 shell_exec('htpasswd -b
/etc/squid3/passwords ' . $username . ' ' . $password);
139
140             }
141 -
142 -             $this->_applyFirewallRules($firewallRules,
'redundant');
143
144 -             if (
145 -                 !empty($this->gatewaysData['data']
['socks']) &&
146 -                 !empty($this->processes['socks']
[0])
147 -             ) {
148 -                 $this->_reconfigure(
149 -                     'socks',
150 -                     'service 3proxy start',
151 -                     '3proxy.cfg',
152 -                     '/usr/local/etc/3proxy/3proxy.pid',
153 -                     0,
154 -                     20,
155 -                     '/usr/local/etc/3proxy/3proxy.cfg',
156 -                     $this->
>gatewaysData['data']['socks']
157
158 -                 );

```

```

130
131 +             if (
132 +                 !empty($this->
>gatewaysData['data']['socks']) &&
133 +                 !empty($this->
>processes['socks'][0])
134 +             ) {
135 +                 $this->_reconfigure(
136 +                     'socks',
137 +                     'service 3proxy
start',
138 +                     '3proxy.cfg',
139 +                     '/usr/local/etc/3proxy/3proxy.pid',
140 +                     0,
141 +                     20,
142 +                     '/usr/local/etc/3proxy/3proxy.cfg',
143 +                     $this->
>gatewaysData['data']['socks']
144 +                 );
145 +             }
146
147
148 +                 $this->_reconfigure(
149 +                     'http',
150 +                     'squid3 start',
151 +                     'squid3',
152 +                     '/var/run/squid3.pid',
153 +                     0,
154 +                     75,
155 +                     '/etc/squid3/squid.conf',
156 +                     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']
['http']['configuration']))
157 +                 );
158 +                 $redundantProcesses = $this->
>processes['http'];
159 +                 unset($redundantProcesses[0]);
160 +                 $redundantProcessChunks =
array_chunk($redundantProcesses,
round(count($redundantProcesses) / 2), true);

```

```

159
160 -         $this->_reconfigure(
161 -             'http',
162 -             'squid3 start',
163 -             'squid3',
164 -             '/var/run/squid3.pid',
165 -             0,
166 -             75,
167 -             '/etc/squid3/squid.conf',
168 -             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']['http']
['configuration']))
169 -         );
170 -         $redundantProcesses = $this-
>processes['http'];
171 -         unset($redundantProcesses[0]);
172 -         $redundantProcessChunks =
array_chunk($redundantProcesses,
round(count($redundantProcesses) / 2), true);
173 -
174 -         foreach ($redundantProcessChunks as
$redundantProcessChunk) {
175 -             $activeRedundantProcesses =
array_keys($redundantProcesses);
176 -             reset($redundantProcessChunk);
177 -             $redundantProcessStart =
key($redundantProcessChunk);
178 -             end($redundantProcessChunk);
179 -             $redundantProcessEnd =
key($redundantProcessChunk);
180 -             $redundantProcessRange =
range($redundantProcessStart, $redundantProcessEnd);
181 -
182 -             foreach ($redundantProcessRange as
$redundantProcess) {
183 -                 unset($activeRedundantProcesses[$redundantProcess - 1]);
184 -             }
185 -
186 -             $activeRedundantPorts = array();

```

```

161 +                 foreach ($redundantProcessChunks
as $redundantProcessChunk) {
162 +                     $activeRedundantProcesses
= array_keys($redundantProcesses);
163 +
164 +                     reset($redundantProcessChunk);
165 +                     $redundantProcessStart =
key($redundantProcessChunk);
166 +                     end($redundantProcessChunk);
167 +                     $redundantProcessEnd =
key($redundantProcessChunk);
168 +                     $redundantProcessRange =
range($redundantProcessStart, $redundantProcessEnd);
169 +
170 +                     foreach
($redundantProcessRange as $redundantProcess) {
171 +                         unset($activeRedundantProcesses[$redundantProcess - 1]);
172 +                     }
173 +
174 +                     $activeRedundantPorts =
array();
175 +
176 +                     foreach

```

```

187
188 -         foreach ($redundantProcesses as
$key => $redundantProcess) {
189 -             if (in_array($key,
$activeRedundantProcesses)) {
190 -
$activeRedundantPorts = array_merge($activeRedundantPorts,
$redundantProcesses[$key]);

```

```

191     }

```

```

192 }

```

```

193

```

```

($redundantProcesses as $key => $redundantProcess) {
176 +             if (in_array($key,
$activeRedundantProcesses)) {
177 +
$activeRedundantPorts = array_merge($activeRedundantPorts,
$redundantProcesses[$key]);
178 +             }
179 +         }
180 +
$overridePorts = array(
181 +             'http' =>
array(array_merge(array(
183 +                 '80',
184 +                 '8888',
185 +                 '55555'
186 +             )),
$activeRedundantPorts),
187 +             'socks' => array(
array(
188 +
189 +                 '1090'
190 +             )
191 +         )
192 +     );
193
194 +     $firewallRules = $this->_configureFirewallRules(false, $overridePorts);
195 +     $this->_applyFirewallRules($firewallRules, 'elastic');
196 +
197 +     foreach
($redundantProcessRange as $redundantProcess) {
198 +         $this->_reconfigure(
199 +             'http',
200 +             'squid3-
redundant' . $redundantProcess . ' start -f
/etc/squid3/squid-redundant' . $redundantProcess .
'.conf',
201 +             'squid3-
redundant' . $redundantProcess,
202 +             '/var/run/squid-redundant' . $redundantProcess . '.pid',
203 +             0,
204 +             0,
205 +             '/etc/squid3/squid-redundant' . $redundantProcess .
'.conf',
206 +             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']['http']
['configuration']))
207 +         );
208 +     }
209 +
210 +     sleep(75);
211 + }
212

```



```

194         $overridePorts = array(
195             -             'http' =>
196                 array(array_merge(array(
197                     -             '80',
198                     -             '8888',
199                     -             '55555'
200                 )),
201                 $activeRedundantPorts),
202             -             'socks' => array(
203                 -             array(
204                     -             '1090'
205                 )
206             );
207         $firewallRules = $this->
208         >_configureFirewallRules(false, $overridePorts);
209         $this->
210         >_applyFirewallRules($firewallRules, 'elastic');
211         foreach ($redundantProcessRange as
212         $redundantProcess) {
213             $this->_reconfigure(
214                 'http',
215                 'squid3-redundant'
216                 . $redundantProcess . ' start -f /etc/squid3/squid-
217                 redundant' . $redundantProcess . '.conf',
218                 'squid3-redundant'
219                 . $redundantProcess,
220                 '/var/run/squid-
221                 redundant' . $redundantProcess . '.pid',
222                 0,
223                 0,
224                 '/etc/squid3/squid-redundant' . $redundantProcess .
225                 '.conf',
226                 str_replace('[pid]', 'pid_filename /var/run/squid-
227                 redundant' . $redundantProcess . '.pid',
228                 str_replace('[ports]', 'http_port ' . implode("\n" .
229                 'http_port ', $this->processes['http']
230                 [$redundantProcess]), $this->gatewaysData['data']['http']
231                 ['configuration']))
232             );
233         }
234         sleep(75);
235     }
236
237     $overridePorts = array(
238         -             'http' => $this->processes['http']
239         );
240     $firewallRules = $this->
241     >_configureFirewallRules(false, $overridePorts);
242     $this->_applyFirewallRules($firewallRules,
243     'elastic');
244
245     if (!empty($this->gatewaysData['data']
246     ['socks-redundant'])) {
247
248         $this->_reconfigure(
249             'socks',
250             'service 3proxy-
251             redundant start',
252             '3proxy-
253             redundant.conf',
254             '/usr/local/etc/3proxy/3proxy-redundant.pid',
255             0,
256             40,
257             '/usr/local/etc/3proxy/3proxy-redundant.conf',
258             $this->
259             >gatewaysData['data']['socks-redundant']
260         );
261     }
262
263     $this->
264     >_applyFirewallRules($firewallRules, 'elastic');
265     unlink('/scripts/pid/reconfigure.pid');
266
267     return true;

```

```

233 -             $this->_reconfigure(
234 -                 'socks',
235 -                 'service 3proxy-redundant
start',
236 -                 '3proxy-redundant.cfg',
237 -
238 -                 '/usr/local/etc/3proxy/3proxy-redundant.pid',
239 -                 0,
240 -                 40,
241 -                 $this->gatewaysData['data']['socks-redundant']
242 -             );
243 -         }
244 -
245 -         $this->_applyFirewallRules($firewallRules,
'elastic');
246 -         unlink('/scripts/pid/reconfigure.pid');
247 -
248 -         return true;
249 -     }
250 -
251 - /**
252 -  * DNS redundancy health checks and process recovery
253 -  *
254 -  * @return boolean
255 -  */
256 -     protected function _checkDNS() {
257 -         if (file_exists('/scripts/pid/dns.pid')) {
258 -             $lastRan =
file_get_contents('/scripts/pid/dns.pid');
259 -
260 -             if ($lastRan < strtotime('-2
minutes')) {
261 -
262 -                 unlink('/scripts/pid/dns.pid');
263 -             } else {
return false;

```

```

236         }
237
238 +     /**
239 +      * DNS redundancy health checks and process
recovery
240 +      *
241 +      * @return boolean
242 +      */
243 +     protected function _checkDNS() {
244 +         if
(file_exists('/scripts/pid/dns.pid')) {
245 +             $lastRan =
file_get_contents('/scripts/pid/dns.pid');
246 +
247 +             if ($lastRan <
strtotime('-2 minutes')) {
248 +
249 +                 unlink('/scripts/pid/dns.pid');
250 +             } else {
return false;
251 +             }
252 +         }
253 +
254 +         file_put_contents('/scripts/pid/dns.pid', time());
255 +         $this->gatewaysData =
json_decode(file_get_contents('/scripts/cache/gatewaysData
'), true);
256 +         array_shift($this->gatewaysData['data']['dns_ips']);
257 +
258 +         if (empty($this->gatewaysData['data']['dns_ips'])) {
return false;
259

```

```

264         }
265     -     }
266     -
267     -     file_put_contents('/scripts/pid/dns.pid',
268     time());
269     -     $this->gatewaysData =
270     json_decode(file_get_contents('/scripts/cache/gatewaysData
271     '), true);
272     -     array_shift($this->gatewaysData['data']
273     ['dns_ips']);
274     -
275     -     if (empty($this->gatewaysData['data']
276     ['dns_ips'])) {
277     -         return false;
278     -     }
279     -
280     -     $dnsIps = array_values($this->
281     gatewaysData['data']['dns_ips']);
282     -
283     -     foreach ($dnsIps as $key => $dnsIp) {
284     -
285     -         $processName = $key == 0 ? 'named'
286     : 'named-redundant' . $key;
287     -         $dnsResponse = array();
288     -         exec('dig +time=2 +tries=1 proxies
289     @' . $dnsIp . ' 2>&1', $dnsResponse);
290     -
291     -         if (
292     -             !empty($dnsResponse[3]) &&
293     -
294     -             strpos(strtolower($dnsResponse[3]), 'got answer') ===
295     false
296     -         ) {
297     -             $dnsProcesses = array();
298     -
299     -             exec('ps $(pgrep named)
300     2>&1', $dnsProcesses);
301     -
302     -             if (!empty($dnsProcesses))
303     {
304     -                 foreach
305     ($dnsProcesses as $dnsProcess) {
306     -
307     -                     $dnsProcess = array_map('strtolower', array_map('trim',
308     array_values(array_filter(explode(' ', $dnsProcess))));
309     -
310     -                     if (
311     -
312     -                     !empty($dnsProcess[0]) &&
313     -
314     -                     is_numeric($dnsProcess[0]) &&
315     -
316     -                     in_array('/usr/sbin/' . $processName, $dnsProcess)
317     ) {
318     -
319     -                     $killProcesses = array();
320     -
321     -                     $shellCommands = array(
322     -
323     -                     '#!' . $this->shell,

```

```

260     }
261
262     +         $dnsIps = array_values($this->
263     gatewaysData['data']['dns_ips']);
264     +
265     +         foreach ($dnsIps as $key =>
266     $dnsIp) {
267     +
268     +             $processName = $key == 0 ?
269     'named' : 'named-redundant' . $key;
270     +             $dnsResponse = array();
271     +             exec('dig +time=2 +tries=1
272     proxies @' . $dnsIp . ' 2>&1', $dnsResponse);
273     +
274     +             if (
275     +
276     +             !empty($dnsResponse[3]) &&
277     +
278     +             strpos(strtolower($dnsResponse[3]), 'got answer') ===
279     false
280     +         ) {
281     +             $dnsProcesses =
282     array();
283     +
284     +             exec('ps $(pgrep
285     named) 2>&1', $dnsProcesses);
286     +
287     +             if
288     (!empty($dnsProcesses)) {
289     +                 foreach
290     ($dnsProcesses as $dnsProcess) {
291     +
292     +                     $dnsProcess = array_map('strtolower', array_map('trim',
293     array_values(array_filter(explode(' ', $dnsProcess))));
294     +
295     +                     if
296     (
297     +
298     +                     !empty($dnsProcess[0]) &&
299     +
300     +                     is_numeric($dnsProcess[0]) &&
301     +
302     +                     in_array('/usr/sbin/' . $processName, $dnsProcess)
303     )
304     +                 {
305     +
306     +                     $killProcesses = array();
307     +
308     +                     $shellCommands = array(
309     +
310     +                     '#!' . $this->shell,

```

```

301 - 'kill -9 ' . trim($dnsProcess[0])
302 - );
303 -
304 - unlink('/scripts/dns.sh');
305 - file_put_contents('/scripts/dns.sh', implode("\n",
306 - $shellCommands));
307 - shell_exec('chmod +x /scripts/dns.sh');
308 - shell_exec('/scripts/./dns.sh');
309 -
310 - }
311 -
312 - shell_exec('service ' .
313 - str_replace('named', 'bind9', $processName) . ' start');
314 -
315 - sleep(1);
316 -
317 - unlink('/scripts/pid/dns.pid');
318 - $this->_checkDNS();
319 -
320 - }
321 -
322 - }
323 -
324 - }
325 -
326 - }
327 -
328 - }
329 -
330 - }
331 -
332 - }
333 -
334 - }
335 -
336 - }
337 -
338 - }
339 -
340 - }
341 -
342 - }
343 -
344 - }
345 -
346 - }
347 -
348 - }
349 -
350 - }
351 -
352 - }
353 -
354 - }
355 -
356 - }
357 -
358 - }
359 -
360 - }
361 -
362 - }
363 -
364 - }
365 -
366 - }
367 -
368 - }
369 -
370 - }
371 -
372 - }
373 -
374 - }
375 -
376 - }
377 -
378 - }
379 -
380 - }
381 -
382 - }
383 -
384 - }
385 -
386 - }
387 -
388 - }
389 -
390 - }
391 -
392 - }
393 -
394 - }
395 -
396 - }
397 -
398 - }
399 -
400 - }
401 -
402 - }
403 -
404 - }
405 -
406 - }
407 -
408 - }
409 -
410 - }
411 -
412 - }
413 -
414 - }
415 -
416 - }
417 -
418 - }
419 -
420 - }
421 -
422 - }
423 -
424 - }
425 -
426 - }
427 -
428 - }
429 -
430 - }
431 -
432 - }
433 -
434 - }
435 -
436 - }
437 -
438 - }
439 -
440 - }
441 -
442 - }
443 -
444 - }
445 -
446 - }
447 -
448 - }
449 -
450 - }
451 -
452 - }
453 -
454 - }
455 -
456 - }
457 -
458 - }
459 -
460 - }
461 -
462 - }
463 -
464 - }
465 -
466 - }
467 -
468 - }
469 -
470 - }
471 -
472 - }
473 -
474 - }
475 -
476 - }
477 -
478 - }
479 -
480 - }
481 -
482 - }
483 -
484 - }
485 -
486 - }
487 -
488 - }
489 -
490 - }
491 -
492 - }
493 -
494 - }
495 -
496 - }
497 -
498 - }
499 -
500 - }
501 -
502 - }
503 -
504 - }
505 -
506 - }
507 -
508 - }
509 -
510 - }
511 -
512 - }
513 -
514 - }
515 -
516 - }
517 -
518 - }
519 -
520 - }
521 -
522 - }
523 -
524 - }
525 -
526 - }
527 -
528 - }
529 -
530 - }
531 -
532 - }
533 -
534 - }
535 -
536 - }
537 -
538 - }
539 -
540 - }
541 -
542 - }
543 -
544 - }
545 -
546 - }
547 -
548 - }
549 -
550 - }
551 -
552 - }
553 -
554 - }
555 -
556 - }
557 -
558 - }
559 -
560 - }
561 -
562 - }
563 -
564 - }
565 -
566 - }
567 -
568 - }
569 -
570 - }
571 -
572 - }
573 -
574 - }
575 -
576 - }
577 -
578 - }
579 -
580 - }
581 -
582 - }
583 -
584 - }
585 -
586 - }
587 -
588 - }
589 -
590 - }
591 -
592 - }
593 -
594 - }
595 -
596 - }
597 -
598 - }
599 -
600 - }
601 -
602 - }
603 -
604 - }
605 -
606 - }
607 -
608 - }
609 -
610 - }
611 -
612 - }
613 -
614 - }
615 -
616 - }
617 -
618 - }
619 -
620 - }
621 -
622 - }
623 -
624 - }
625 -
626 - }
627 -
628 - }
629 -
630 - }
631 -
632 - }
633 -
634 - }
635 -
636 - }
637 -
638 - }
639 -
640 - }
641 -
642 - }
643 -
644 - }
645 -
646 - }
647 -
648 - }
649 -
650 - }
651 -
652 - }
653 -
654 - }
655 -
656 - }
657 -
658 - }
659 -
660 - }
661 -
662 - }
663 -
664 - }
665 -
666 - }
667 -
668 - }
669 -
670 - }
671 -
672 - }
673 -
674 - }
675 -
676 - }
677 -
678 - }
679 -
680 - }
681 -
682 - }
683 -
684 - }
685 -
686 - }
687 -
688 - }
689 -
690 - }
691 -
692 - }
693 -
694 - }
695 -
696 - }
697 -
698 - }
699 -
700 - }
701 -
702 - }
703 -
704 - }
705 -
706 - }
707 -
708 - }
709 -
710 - }
711 -
712 - }
713 -
714 - }
715 -
716 - }
717 -
718 - }
719 -
720 - }
721 -
722 - }
723 -
724 - }
725 -
726 - }
727 -
728 - }
729 -
730 - }
731 -
732 - }
733 -
734 - }
735 -
736 - }
737 -
738 - }
739 -
740 - }
741 -
742 - }
743 -
744 - }
745 -
746 - }
747 -
748 - }
749 -
750 - }
751 -
752 - }
753 -
754 - }
755 -
756 - }
757 -
758 - }
759 -
760 - }
761 -
762 - }
763 -
764 - }
765 -
766 - }
767 -
768 - }
769 -
770 - }
771 -
772 - }
773 -
774 - }
775 -
776 - }
777 -
778 - }
779 -
780 - }
781 -
782 - }
783 -
784 - }
785 -
786 - }
787 -
788 - }
789 -
790 - }
791 -
792 - }
793 -
794 - }
795 -
796 - }
797 -
798 - }
799 -
800 - }
801 -
802 - }
803 -
804 - }
805 -
806 - }
807 -
808 - }
809 -
810 - }
811 -
812 - }
813 -
814 - }
815 -
816 - }
817 -
818 - }
819 -
820 - }
821 -
822 - }
823 -
824 - }
825 -
826 - }
827 -
828 - }
829 -
830 - }
831 -
832 - }
833 -
834 - }
835 -
836 - }
837 -
838 - }
839 -
840 - }
841 -
842 - }
843 -
844 - }
845 -
846 - }
847 -
848 - }
849 -
850 - }
851 -
852 - }
853 -
854 - }
855 -
856 - }
857 -
858 - }
859 -
860 - }
861 -
862 - }
863 -
864 - }
865 -
866 - }
867 -
868 - }
869 -
870 - }
871 -
872 - }
873 -
874 - }
875 -
876 - }
877 -
878 - }
879 -
880 - }
881 -
882 - }
883 -
884 - }
885 -
886 - }
887 -
888 - }
889 -
890 - }
891 -
892 - }
893 -
894 - }
895 -
896 - }
897 -
898 - }
899 -
900 - }
901 -
902 - }
903 -
904 - }
905 -
906 - }
907 -
908 - }
909 -
910 - }
911 -
912 - }
913 -
914 - }
915 -
916 - }
917 -
918 - }
919 -
920 - }
921 -
922 - }
923 -
924 - }
925 -
926 - }
927 -
928 - }
929 -
930 - }
931 -
932 - }
933 -
934 - }
935 -
936 - }
937 -
938 - }
939 -
940 - }
941 -
942 - }
943 -
944 - }
945 -
946 - }
947 -
948 - }
949 -
950 - }
951 -
952 - }
953 -
954 - }
955 -
956 - }
957 -
958 - }
959 -
960 - }
961 -
962 - }
963 -
964 - }
965 -
966 - }
967 -
968 - }
969 -
970 - }
971 -
972 - }
973 -
974 - }
975 -
976 - }
977 -
978 - }
979 -
980 - }
981 -
982 - }
983 -
984 - }
985 -
986 - }
987 -
988 - }
989 -
990 - }
991 -
992 - }
993 -
994 - }
995 -
996 - }
997 -
998 - }
999 -
1000 - }

```

```

321
322 - /**
323 - * Check HTTP and SOCKS ports
324 - *
325 - * @param string $ip Proxy IP
326 - * @param string $port Proxy port
327 - * @param string $port Proxy protocol (http or socks)
328 - * @param integer $integer Request timeout
329 - *
330 - * @return boolean $alive True if port is active, false
331 if refusing connections
332 - */
333 - protected function _checkPort($ip, $port,
334 $protocol, $timeout = 5) {
335 -     $alive = false;
336 -
337 -     switch ($protocol) {
338 -         case 'http':
339 -             $response =
340 shell_exec('curl -I -s -x ' . $ip . ':' . $port . '
341 http://squid -v --connect-timeout ' . $timeout . ' --max-
342 time ' . $timeout);
343 -
344 -             if ($this->_strposa(strtolower($response), array(
345 '407 proxy',
346 '403 forbidden',
347 ' 503 ',
348 ' timed out '
349 )) !== false) {
350 -                 $alive = true;
351 -             }
352 -             break;
353 -         case 'socks':
354 -             exec('curl --socks5-
355 hostname ' . $ip . ':' . $port . ' http://socks/ -v --
356 connect-timeout ' . $timeout . ' --max-time ' . $timeout .
357 ' 2>&1', $socksResponse);
358 -             $socksResponse =
359 end($socksResponse);
360 -             $alive =
361 (strpos(strtolower($socksResponse), 'empty reply ') !==
362 false);
363 -             break;
364 -         }
365 -     }
366 -     return $alive;
367 - }

```

```

330 +         ' timed
331 out '
332 +             )) !== false) {
333 +                 $alive =
334 true;
335 +             }
336 +             break;
337 +         case 'socks':
338 +             exec('curl --
339 socks5-hostname ' . $ip . ':' . $port . ' http://socks/ -v
340 --connect-timeout ' . $timeout . ' --max-time ' . $timeout
341 . ' 2>&1', $socksResponse);
342 +             $socksResponse =
343 end($socksResponse);
344 +             $alive =
345 (strpos(strtolower($socksResponse), 'empty reply ') !==
346 false);
347 +             break;
348 -         }
349 +     }
350 +     return $alive;
351 + }
352 +
353 + /**
354 + * Configure firewall rules
355 + *
356 + * @param boolean $redundant Route to redundant

```

```

358
359 - /**
360 - * Configure firewall rules
361 - *
362 - * @param boolean $redundant Route to redundant process
ports only if true, all ports if false
363 - * @param array $overridePorts Override default base
ports
364 - *
365 - * @return array $rules Firewall rules
366 - */
367 - protected function
_configureFirewallRules($redundant = false, $overridePorts
= array()) {
368 -     $rules = array(
369 -         '*filter',
370 -         ':INPUT ACCEPT [0:0]',
371 -         ':FORWARD ACCEPT [0:0]',
372 -         ':OUTPUT ACCEPT [0:0]',
373 -         '-A INPUT -p icmp -m hashlimit --
hashlimit-name icmp --hashlimit-mode srcip --hashlimit
1/second --hashlimit-burst 2 -j ACCEPT'
374 -     );
375 -
376 -     if (
377 -         !empty($this->sshPorts) &&
378 -         is_array($this->sshPorts)
379 -     ) {
380 -         foreach ($this->sshPorts as
$sshPort) {
381 -             if (is_numeric($sshPort))
{
382 -                 $rules[] = '-A
INPUT -p tcp -m tcp --dport ' . $sshPort . ' -m connlimit
--connlimit-above 4 --connlimit-mask 32 --connlimit-saddr
-j DROP';
383 -                 $rules[] = '-A

```

```

INPUT -p tcp -m tcp --dport ' . $sshPort . ' -m hashlimit
--hashlimit-upto 15/hour --hashlimit-burst 3 --hashlimit-
mode srcip --hashlimit-name ssh --hashlimit-htable-expire
500000 -j ACCEPT';
384         }
385     }
386 -     }
387
388 -     if (
389 -         !empty($this->gatewaysData['data']
390 -             ['firewall_filter']) &&
391 -         is_array($this-
392 -             >gatewaysData['data']['firewall_filter'])
393 -         ) {
394 -             foreach ($this-
395 -             >gatewaysData['data']['firewall_filter'] as $rule) {
396 -                 $rules[] = $rule;
397 -             }
398 -         }
399
400 -     if (
401 -         !empty($this->gatewaysData['data']
402 -             ['dns_ips']) &&
403 -         is_array($this-
404 -             >gatewaysData['data']['dns_ips'])
405 -         ) {
406 -             $rules[] = '-A OUTPUT -d ' .
407 -             implode(',', $this->gatewaysData['data']['dns_ips']) . ' -
408 -             p udp -m udp -j ACCEPT';
409 -         }
410
411 -         $rules[] = '-A OUTPUT -s 127.0.0.0/24 -p
412 -         udp -j DROP';
413 -         $rules[] = 'COMMIT';
414 -         $rules[] = '*nat';
415 -         $rules[] = ':PREROUTING ACCEPT [0:0]';
416
417 -         $rules[] = ':INPUT ACCEPT [0:0]';
418 -         $rules[] = ':OUTPUT ACCEPT [0:0]';
419 -         $rules[] = ':POSTROUTING ACCEPT [0:0]';
420
421 -         $dnsIps = array_values($this-
422 -             >gatewaysData['data']['dns_ips']);
423 -         unset($dnsIps[0]);
424 -         krsort($dnsIps);
425
426 -         foreach ($dnsIps as $key => $dnsIp) {
427 -             $loadBalancer = '';
428
429 -             if ($key > 1) {
430 -                 $loadBalancer = '-m
431 -                 statistic --mode nth --every ' . $key . ' --packet 0 ';
432 -             }
433
434 -             $rules[] = '-A OUTPUT -d
435 -             127.0.0.1/32 -p udp -m udp --dport 53 ' . $loadBalancer .
436 -             '-j DNAT --to-destination ' . $dnsIp;
437 -         }
438     }
439 }
440 }
441

```

```

372     }
373 }
374
375 +     if (
376 +         !empty($this-
377 +             >gatewaysData['data']['firewall_filter']) &&
378 +         is_array($this-
379 +             >gatewaysData['data']['firewall_filter'])
380 +         ) {
381 +             foreach ($this-
382 +             >gatewaysData['data']['firewall_filter'] as $rule) {
383 +                 $rules[] = $rule;
384 +             }
385 -         }
386
387 +     if (
388 +         !empty($this-
389 +             >gatewaysData['data']['dns_ips']) &&
390 +         is_array($this-
391 +             >gatewaysData['data']['dns_ips'])
392 +         ) {
393 +             $rules[] = '-A OUTPUT -d '
394 +             . implode(',', $this->gatewaysData['data']['dns_ips']) . ' -
395 +             p udp -m udp -j ACCEPT';
396 +         }
397
398 +         $rules[] = '-A OUTPUT -s
399 +         127.0.0.0/24 -p udp -j DROP';
400 +         $rules[] = 'COMMIT';
401 +         $rules[] = '*nat';
402 +         $rules[] = ':PREROUTING ACCEPT
403 +         [0:0]';
404
405 +         $rules[] = ':INPUT ACCEPT [0:0]';
406 +         $rules[] = ':OUTPUT ACCEPT [0:0]';
407 +         $rules[] = ':POSTROUTING ACCEPT
408 +         [0:0]';
409
410 +         $dnsIps = array_values($this-
411 +             >gatewaysData['data']['dns_ips']);
412 +         unset($dnsIps[0]);
413 +         krsort($dnsIps);
414
415 +         foreach ($dnsIps as $key =>
416 +             $dnsIp) {
417 +             $loadBalancer = '';
418
419 +             if ($key > 1) {
420 +                 $loadBalancer = '-m
421 +                 statistic --mode nth --every ' . $key . ' --packet 0 ';
422 +             }
423
424 +             $rules[] = '-A OUTPUT -d
425 +             127.0.0.1/32 -p udp -m udp --dport 53 ' . $loadBalancer .
426 +             '-j DNAT --to-destination ' . $dnsIp;
427 +         }
428     }
429 }
430 }
431

```

```

422 -
423 -             $rules[] = '-A OUTPUT -d
127.0.0.1/32 -p udp -m udp --dport 53 ' . $loadBalancer .
'-j DNAT --to-destination ' . $dnsIp;
424 -         }

425
426 -         if (
427 -             !empty($this->processes) &&
428 -             is_array($this->processes)
429 -         ) {
430 -             foreach ($this->processes as
$protocol => $processes) {
431 -                 $basePorts = array();
432 -
433 -                 if ($redundant) {
434 -                     unset($processes[0]);
435 -                     $processes =
array_values($processes);
436 -                 }
437 -
438 -                 if
(!empty($overridePorts)) {
439 -                     if
(empty($overridePorts[$protocol])) {
440 -                         continue;
441 -                     }
442 -
443 -                     $processes =
$overridePorts[$protocol];
444 -                 }
445 -
446 -                 foreach ($processes as
$ports) {
447 -                     $basePorts =
array_merge($ports, $basePorts);
448 -                 }
449 -
450 -                 $ports =
array_unique($basePorts);
451 -
452 -                 foreach ($ports as $key =>
$port) {
453 -                     if ($this->
_checkPort($this->gatewaysData['data']['proxy_ips'][0],
$port, $protocol) === false) {
454 -                         unset($ports[$key]);
455 -                     }
456 -                 }
457 -             }

```

```

412 +
413 +             if (
414 +                 !empty($this->processes)
&&
415 +                 is_array($this->processes)
416 +             ) {
417 +                 foreach ($this->processes
as $protocol => $processes) {
418 +                     $basePorts =
array();
419 +
420 +                     if ($redundant) {
421 +                         unset($processes[0]);
422 +                         $processes
= array_values($processes);
423 +                     }
424 +
425 +                     if
(!empty($overridePorts)) {
426 +                         if
(empty($overridePorts[$protocol])) {
427 +                             continue;
428 +                         }
429 +
430 +                         $processes
= $overridePorts[$protocol];
431 +                     }
432 +
433 +                     foreach
($processes as $ports) {
434 +                         $basePorts
= array_merge($ports, $basePorts);
435 +                     }
436 +
437 +                     $ports =
array_unique($basePorts);
438 +
439 +                     foreach ($ports as
$key => $port) {
440 +                         if ($this->
_checkPort($this->gatewaysData['data']['proxy_ips'][0],
$port, $protocol) === false) {
441 +                             unset($ports[$key]);
442 +                         }
443 +                     }
444 +

```



```

458 - shuffle($ports);
459 - $ports =
array_values($ports);
460 - krsort($ports);
461 - $dports =
array_merge($this->processes[$protocol][0], $basePorts);
462 - $dportsSplit =
array_chunk($dports, '5');
463
464 - if (!empty($ports)) {
465 -         foreach
($dportsSplit as $chunk => $dports) {
466 -             $dports =
implode(',', $dports);
467
468 -         foreach
($ports as $key => $port) {
469 -             $loadBalancer = '';
470
471 -             if
($key > 0) {
472 -                 $loadBalancer = '-m statistic --mode nth --every ' . ($key
+ 1) . ' --packet 0 ';
473 -             }
474
475 -             $rules[] = '-A PREROUTING -p tcp -m multiport --dports ' .
$dports . ' ' . $loadBalancer . '-j DNAT --to-destination
:' . $port . ' --persistent';
476 -             }
477 -         }
478 -     }
479 - }
480 - }
481 -
482 - $rules[] = 'COMMIT';
483 - $rules = array_chunk($rules, 100);
484 - return $rules;
485 - }
486
487 - /**
488 -  * Create writable log and cache directories
489 -  *
490 -  * @return
491 -  */
492 - protected function _createDirectories() {
493 -     $directories = array(
494 -         '/scripts',
495 -         '/scripts/cache',
496 -         '/scripts/errors',
497 -         '/scripts/iptables',
498 -         '/scripts/pid'
499 -     );
500 -
501 -     foreach ($directories as $directory) {
502 -         if (!is_dir($directory)) {
503 -             shell_exec('mkdir -m 777 '
. $directory);

```

```

504 -         }
505 -     }
506
507 -         return;
508 -     }
509 -
510 - /**
511 -  * Get process IDs from process name
512 -  *
513 -  * @param string $processName Process name
514 -  * @param string $configurationFile Full path to process
configuration file
515 -  *
516 -  * @return array $processIds Process IDs
517 -  */
518 -     protected function _getProcessIds($processName,
$configurationFile) {
519 -         $processIds = array();
520 -         exec('ps -fC ' . $processName . ' 2>&1',
$processes);
521 -
522 -         if (!empty($processes[0])) {
523 -             unset($processes[0]);
524 -
525 -             foreach ($processes as $process) {
526 -                 $processColumns =
array_values(array_filter(explode(' ', $process)));
527
528 -                 if (
529 -
!empty($processColumns[1]) &&
530 -
(
531 -
empty($processes[2]) ||
532 -
strpos($configurationFile, '-redundant') === false ||
533 -
$this->_strposa($process, array(
534 -
$processName . ' ',
535 -
$configurationFile
536 -
)) !==
false
537 -
)
538 -
) {
539 -
$processIds[] =
$processColumns[1];
540 -
}
541 -
}
542 -
}
543 -
544 -     return $processIds;
545 - }

472 -     }
473
474 +     /**
475 +      * Create writable log and cache directories
476 +      *
477 +      * @return
478 +      */
479 +     protected function _createDirectories() {
480 +         $directories = array(
481 +             '/scripts',
482 +
'/scripts/cache',
483 +
'/scripts/errors',
484 +
'/scripts/iptables',
485 +
'/scripts/pid'
486 +
);
487
488 +     foreach ($directories as
$directory) {
489 +         if (!is_dir($directory)) {
490 +             shell_exec('mkdir
-m 777 ' . $directory);
491 +
}
492 +
}
493 +
494 +     return;
495 +
}
496
497 +     /**
498 +      * Get process IDs from process name
499 +      *
500 +      * @param string $processName Process name
501 +      * @param string $configurationFile Full path to
process configuration file
502 +      *

```

```

503 +         * @return array $processIds Process IDs
504 +         */
505 +         protected function
506 _getProcessIds($processName, $configurationFile) {
507 +             $processIds = array();
508 +             exec('ps -fC ' . $processName . '
509 2>&1', $processes);
510 +
511 +             if (!empty($processes[0])) {
512 +                 unset($processes[0]);
513 +
514 +                 foreach ($processes as
515 $process) {
516 +                     $processColumns =
517 array_values(array_filter(explode(' ', $process)));
518 +
519 +                     if (
520 !empty($processColumns[1]) &&
521 (
522 empty($processes[2]) ||
523 strpos($configurationFile, '-redundant') === false ||
524 $this->_strposa($process, array(
525 $processName . ' ',
526 $configurationFile
527 ))
528 !== false
529 )
530 ) {
531 +                         $processIds[] = $processColumns[1];
532 +                     }
533 +                 }
534 +             }
535 +
536 +             return $processIds;
537 +         }
538 +     }
539 + }
540 + }
541 + }
542 + }
543 + }
544 + }
545 + }
546 + }
547 + }
548 + }
549 + }
550 + }
551 + }
552 + }
553 + }
554 + }
555 + }
556 + }
557 + }
558 + }
559 + }
560 + }
561 + }
562 + }
563 + }
564 + }
565 + }
566 + }
567 + }
568 + }
569 + }
570 + }
571 + }
572 + }
573 + }
574 + }
575 + }
576 + }
577 + }
578 + }
579 + }
580 + }
581 + }
582 + }
583 + }
584 + }
585 + }
586 + }
587 + }
588 + }
589 + }
590 + }
591 + }
592 + }
593 + }
594 + }
595 + }
596 + }
597 + }
598 + }
599 + }
600 + }
601 + }
602 + }
603 + }
604 + }
605 + }
606 + }
607 + }
608 + }
609 + }
610 + }
611 + }
612 + }
613 + }
614 + }
615 + }
616 + }
617 + }
618 + }
619 + }
620 + }
621 + }
622 + }
623 + }
624 + }
625 + }
626 + }
627 + }
628 + }
629 + }
630 + }
631 + }
632 + }
633 + }
634 + }
635 + }
636 + }
637 + }
638 + }
639 + }
640 + }
641 + }
642 + }
643 + }
644 + }
645 + }
646 + }
647 + }
648 + }
649 + }
650 + }
651 + }
652 + }
653 + }
654 + }
655 + }
656 + }
657 + }
658 + }
659 + }
660 + }
661 + }
662 + }
663 + }
664 + }
665 + }
666 + }
667 + }
668 + }
669 + }
670 + }
671 + }
672 + }
673 + }
674 + }
675 + }
676 + }
677 + }
678 + }
679 + }
680 + }
681 + }
682 + }
683 + }
684 + }
685 + }
686 + }
687 + }
688 + }
689 + }
690 + }
691 + }
692 + }
693 + }
694 + }
695 + }
696 + }
697 + }
698 + }
699 + }
700 + }
701 + }
702 + }
703 + }
704 + }
705 + }
706 + }
707 + }
708 + }
709 + }
710 + }
711 + }
712 + }
713 + }
714 + }
715 + }
716 + }
717 + }
718 + }
719 + }
720 + }
721 + }
722 + }
723 + }
724 + }
725 + }
726 + }
727 + }
728 + }
729 + }
730 + }
731 + }
732 + }
733 + }
734 + }
735 + }
736 + }
737 + }
738 + }
739 + }
740 + }
741 + }
742 + }
743 + }
744 + }
745 + }
746 + }
747 + }
748 + }
749 + }
750 + }
751 + }
752 + }
753 + }
754 + }
755 + }
756 + }
757 + }
758 + }
759 + }
760 + }
761 + }
762 + }
763 + }
764 + }
765 + }
766 + }
767 + }
768 + }
769 + }
770 + }
771 + }
772 + }
773 + }
774 + }
775 + }
776 + }
777 + }
778 + }
779 + }
780 + }
781 + }
782 + }
783 + }
784 + }
785 + }
786 + }
787 + }
788 + }
789 + }
790 + }
791 + }
792 + }
793 + }
794 + }
795 + }
796 + }
797 + }
798 + }
799 + }
800 + }
801 + }
802 + }
803 + }
804 + }
805 + }
806 + }
807 + }
808 + }
809 + }
810 + }
811 + }
812 + }
813 + }
814 + }
815 + }
816 + }
817 + }
818 + }
819 + }
820 + }
821 + }
822 + }
823 + }
824 + }
825 + }
826 + }
827 + }
828 + }
829 + }
830 + }
831 + }
832 + }
833 + }
834 + }
835 + }
836 + }
837 + }
838 + }
839 + }
840 + }
841 + }
842 + }
843 + }
844 + }
845 + }
846 + }
847 + }
848 + }
849 + }
850 + }
851 + }
852 + }
853 + }
854 + }
855 + }
856 + }
857 + }
858 + }
859 + }
860 + }
861 + }
862 + }
863 + }
864 + }
865 + }
866 + }
867 + }
868 + }
869 + }
870 + }
871 + }
872 + }
873 + }
874 + }
875 + }
876 + }
877 + }
878 + }
879 + }
880 + }
881 + }
882 + }
883 + }
884 + }
885 + }
886 + }
887 + }
888 + }
889 + }
890 + }
891 + }
892 + }
893 + }
894 + }
895 + }
896 + }
897 + }
898 + }
899 + }
900 + }
901 + }
902 + }
903 + }
904 + }
905 + }
906 + }
907 + }
908 + }
909 + }
910 + }
911 + }
912 + }
913 + }
914 + }
915 + }
916 + }
917 + }
918 + }
919 + }
920 + }
921 + }
922 + }
923 + }
924 + }
925 + }
926 + }
927 + }
928 + }
929 + }
930 + }
931 + }
932 + }
933 + }
934 + }
935 + }
936 + }
937 + }
938 + }
939 + }
940 + }
941 + }
942 + }
943 + }
944 + }
945 + }
946 + }
947 + }
948 + }
949 + }
950 + }
951 + }
952 + }
953 + }
954 + }
955 + }
956 + }
957 + }
958 + }
959 + }
960 + }
961 + }
962 + }
963 + }
964 + }
965 + }
966 + }
967 + }
968 + }
969 + }
970 + }
971 + }
972 + }
973 + }
974 + }
975 + }
976 + }
977 + }
978 + }
979 + }
980 + }
981 + }
982 + }
983 + }
984 + }
985 + }
986 + }
987 + }
988 + }
989 + }
990 + }
991 + }
992 + }
993 + }
994 + }
995 + }
996 + }
997 + }
998 + }
999 + }
1000 + }

```

```

563 -         sleep($delayStart);
564 -     }
565 -
566 -     $killProcesses = array();
567 -     $shellCommands = array(
568 -         '#!' . $this->shell
569 -     );
570 -
571 -     $killProcesses = $this->
572 >_getProcessIds($processName, $configurationFile);
573 -
574 -     foreach ($killProcesses as $killProcess) {
575 -         $shellCommands[] = 'kill -9 ' .
576 trim($killProcess);
577 -     }
578 -
579 -     if (count($shellCommands > 1)) {
580 -         unlink('/scripts/' . $protocol .
581 '.sh');
582 -         file_put_contents('/scripts/' .
583 $protocol . '.sh', implode("\n", $shellCommands));
584 -         shell_exec('chmod +x /scripts/' .
585 $protocol . '.sh');
586 -         shell_exec('/scripts/./' .
587 $protocol . '.sh');
588 -     }
589 -
590 -     if (
591         !empty($configurationFile) &&
592         !empty($configurationData)
593     ) {
594 -         file_put_contents($configurationFile, $configurationData);
595 -     }

```

```

532     }
533 -
534 + /**
535 +  * Reconfigure specific process
536 +  *
537 +  * @param string $protocol Proxy protocol (http or
538 socks)
539 +  * @param string $startCommand Command to start
540 process
541 +  * @param string $processName Process name
542 +  * @param string $processId Full path to process
543 ID
544 +  * @param integer $delayStart Delay at beginning
545 of reconfiguration
546 +  * @param integer $delayEnd Delay at end of
547 reconfiguration
548 +  * @param string $configurationFile Full path to
549 process configuration file
550 +  * @param string $configurationData Process
551 configuration data
552 +  *
553 +  * @return
554 +  */
555 + protected function _reconfigure($protocol,
556 $startCommand, $processName, $processId, $delayStart = 0,
557 $delayEnd = 0, $configurationFile, $configurationData) {
558 +     if ($delayStart) {
559 +         sleep($delayStart);
560 +     }
561 +
562 +     $killProcesses = array();
563 +     $shellCommands = array(
564 +         '#!' . $this->shell
565 +     );
566 +     $killProcesses = $this->
567 >_getProcessIds($processName, $configurationFile);
568 +
569 +     foreach ($killProcesses as
570 $killProcess) {
571 +         $shellCommands[] = 'kill
572 -9 ' . trim($killProcess);
573 +     }
574 +
575 +     if (count($shellCommands > 1)) {
576 +         unlink('/scripts/' .
577 $protocol . '.sh');
578 +         file_put_contents('/scripts/' . $protocol . '.sh',
579 implode("\n", $shellCommands));
580 +         shell_exec('chmod +x
581 /scripts/' . $protocol . '.sh');
582 +         shell_exec('/scripts/./' .
583 $protocol . '.sh');
584 +     }
585 +
586 + }
587 -
588 - }
589 -

```

```

590 -         unlink($processId);
591 -         sleep(2);
592 -         shell_exec($startCommand);
593
594 -         if ($delayEnd) {
595 -             sleep($delayEnd);
596 -         }
597 -     }
598
599 - /**
600 -  * Initiate processes
601 -  *
602 -  * @param string $processName Process name
603 -  *
604 -  * @return boolean $status
605 -  */
606 -     public function start($processName) {
607 -         switch ($processName) {
608 -             case 'apply_firewall':
609 -                 $status = $this->_applyFirewall();
610 -                 break;
611 -             case 'apply_reconfiguration':
612 -                 $status = $this->_applyReconfiguration();
613 -                 break;
614 -             case 'check_dns':
615 -                 $status = $this->_checkDNS();
616 -                 break;
617 -         }
618
619 -         return $status;
620 -     }

```

```

570 +         if (
571 +             !empty($configurationFile)
572 +             &&
573 +             !empty($configurationData)
574 +         ) {
575 +             file_put_contents($configurationFile, $configurationData);
576 +         }
577 +         unlink($processId);
578 +         sleep(2);
579 +         shell_exec($startCommand);
580
581 +         if ($delayEnd) {
582 +             sleep($delayEnd);
583 +         }
584 +     }
585
586 +     /**
587 +      * Initiate processes
588 +      *
589 +      * @param string $processName Process name
590 +      *
591 +      * @return boolean $status
592 +      */
593 +     public function start($processName) {
594 +         switch ($processName) {
595 +             case 'apply_firewall':
596 +                 $status = $this->_applyFirewall();
597 +                 break;
598 +             case 'apply_reconfiguration':
599 +                 $status = $this->_applyReconfiguration();
600 +                 break;
601 +             case 'check_dns':
602 +                 $status = $this->_checkDNS();
603 +                 break;
604 +         }
605 +
606 +         return $status;
607 +     }

```

```

621
622 - /**
623 -  * Format strpos to use array as needle
624 -  *
625 -  * @param array $haystack
626 -  * @param array $needles
627 -  * @param integer $offset
628 -  *
629 -  * @return boolean True if match is found, false if no
match
630 - */
631 -     protected function _strposa($haystack, $needles,
$offset = 0) {
632 -         if (!is_array($needles)) {
633 -             $needles = array($needles);
634 -
635 -         };
636 -         foreach ($needles as $needle) {
637 -             if (strpos($haystack, $needle,
$offset) !== false) {
638 -                 return true;
639 -             }
640 -         }
641 -
642 -         return false;
643 -     }
644 - }
645 - }
646 - ?>

```

```

608
609 +     /**
610 +     * Format strpos to use array as needle
611 +     *
612 +     * @param array $haystack
613 +     * @param array $needles
614 +     * @param integer $offset
615 +     *
616 +     * @return boolean True if match is found, false
if no match
617 +     */
618 +     protected function _strposa($haystack,
$needles, $offset = 0) {
619 +         if (!is_array($needles)) {
620 +             $needles =
array($needles);
621 +         };
622 +
623 +         foreach ($needles as $needle) {
624 +             if (strpos($haystack,
$needle, $offset) !== false) {
625 +                 return true;
626 +             }
627 +         }
628 +
629 +         return false;
630 +     }
631 + }
632 + }
633 - ?>

```

0 comments on commit `37e75ec`