# SwitchON и SwitchOFF (API для бота и биллинга )

|  |
| --- |
| **abonentsService.js (SwitchOn & SwitchOff)** |
| const { spawn } = require('child\_process')  function ipfw(*args*) {    const child = spawn('/sbin/ipfw', *args*)    child.stdout.on('data', (*data*) => {      console.log(`stdout: ${*data*}`)    })    child.stderr.on('data', (*data*) => {      console.error(`stderr: ${*data*}`)    })    child.on('close', (*code*) => {      console.log(`child process exited with code ${*code*}`)    })    return child  }  module.exports.switchOff = async function (*abonentId*, *ipAddress*, *vlanId*) {    try {      const deleteCommand = ipfw(['delete', *abonentId*])      deleteCommand.on('close', (*code*) => {        if (*code* === 0) {          console.log(`Deleted rule ${*abonentId*} successfully.`)        }      })      const addCommand = ipfw(['add', *abonentId*, 'deny', 'ip', 'from', `${*ipAddress*}/32`, 'to', 'any', 'via', `vlan${*vlanId*}`, 'in'])      addCommand.on('close', (*code*) => {        if (*code* === 0) {          console.log(`Added rule ${*abonentId*} successfully.`)        }      })      return true    } catch (error) {      console.error('Error executing commands:', error)      return false    }  }  module.exports.switchOn = async function (*abonentId*) {    try {      const deleteCommand = ipfw(['delete', *abonentId*])      deleteCommand.on('close', (*code*) => {        if (*code* === 0) {          console.log(`Deleted rule ${*abonentId*} successfully.`)        }      })      return true    } catch (error) {      console.error('Error executing commands:', error)      return false    }  }  *//*await abonentsService.switchOff('9999', '192.168.199.199', '199')  *//temprorary for testing*  *//*await abonentsService.switchOn('9999')  *//temprorary for testing* |

В тесте на FreeBSD пришлось убрать module.exports и await abonentsService

**const addTag = process.env.PLATFORM !== 'freebsd' ? 'Test mode' : ''**

## **Forward -on и Forward -off**

|  |
| --- |
| # Создайте правило для перенаправления  sudo ipfw add 100 fwd 91.220.106.9 tcp from 192.168.55.65 to any  # Добавьте правило для перенаправления на конкретную страницу (например, /redirect.html)  sudo ipfw add 101 fwd 91.220.106.9,80 tcp from 192.168.55.65 to any dst-port 80 |
| <VirtualHost \*:80>  ServerName 91.220.106.9  DocumentRoot /путь\_к\_каталогу\_с\_файлами  RewriteEngine On  RewriteRule ^/redirect\.html$ /нужный\_путь\_к\_странице [R,L]  </VirtualHost>  **<VirtualHost \*:7878>**  **DocumentRoot "${SRVROOT}/htdocs"**  **ServerName 127.0.0.1**  **Redirect / http://127.0.0.1:80/redirect.html**  **</VirtualHost>** |
| sudo ipfw table 1 add 192.168.55.65  sudo ipfw table 1 add 192.168.55.66 |
| # Перенаправляем трафик из таблицы 1 на IP-адрес 91.220.106.9  sudo ipfw add 100 fwd 91.220.106.9 tcp from table(1) to any  # Перенаправляем трафик из таблицы 1 на IP-адрес 91.220.106.9 и порт 80  sudo ipfw add 101 fwd 91.220.106.9,80 tcp from table(1) to any dst-port 80 |

# Redirect.

Для таблицы IP адресов неплательщиков делаем редирект на сайт Silver-Service (на сгенерированную страницу)

npm install fastify@latest

|  |
| --- |
| **PORT = 8000**  **PORT\_FOR\_REDIRECT = 8001**  **UPSTREAM\_URL = http://www.silver-service.com.ua/page8629852.html**  **PLATFORM = windows** |
| require('dotenv').config()  const { app, redirectServer } = require('./index')  app.listen({ port: process.env.PORT || 8080 }, (*err*, *address*) => {    if (*err*) {      app.log.error(*err*)      process.exit(1)    }    console.log(`[API] Service listening on ${*address*}`)  })  redirectServer.listen({ port: process.env.PORT\_FOR\_REDIRECT || 8081 }, (*err*, *address*) => {    if (*err*) {      redirectServer.log.error(*err*)      console.error(*err*)    }    console.log(`[Redirect] Service listening on ${*address*}`)  }) |
| const Fastify = require('fastify')  const authPlugin = require('./plugins/auth.plugin')  const httpProxy = require('@fastify/http-proxy')  const app = Fastify({    trustProxy: true  })  const redirectServer = Fastify({    trustProxy: true  })  redirectServer.register(httpProxy, {    upstream: process.env.UPSTREAM\_URL,    prefix: '/',    http2: false  })  redirectServer.all('/redirect', async (*request*, *reply*) => {    try {      const proxyResponse = await redirectServer.proxy(*request*.raw)  *reply*.send(proxyResponse)    } catch (error) {      console.error(error)    }  })  app.register(authPlugin)  app.register(require('./routes/abonents.js'), { prefix: '/api' })  module.exports = { app, redirectServer } |
| [**http://127.0.0.1:8001/**](http://127.0.0.1:8001/) **из браузера - test** |

# Сбор логов (статистики) , обработка и запись на FireStore &| database.

3.1. Сбор логов (статистики)

|  |  |
| --- | --- |
| ~~pkg install pftop~~ | |
| ~~pkg install pflogsumm~~ | |
| [**https://linuxhint.com/awk-command-examples/**](https://linuxhint.com/awk-command-examples/) | |
| **pfctl -ss | awk '{ print $3 "\t" $5 "\t" $6 }' | awk -F ':' '/^192.168./ {print $1}' | sort | uniq -c>> /usr/home/ftp1c/stat168.log**  **date -j +%Y-%m-%d\ %H:%M**  for bash  **timestamp=$(date "+%Y-%m-%d %H:%M")**  **echo $timestamp** | |
| **pfctl -ss | awk '{ print $4 "\t" $6 "\t" $7 }'** | (192.168.111.84:60306) 34.195.171.183:443 ESTABLISHED:ESTABLISHED  (192.168.19.81:19150) 84.16.241.132:63142 0:0  (192.168.29.142:55772) 46.175.244.67:12275 TIME\_WAIT:TIME\_WAIT  (192.168.29.140:53022) 142.250.179.138:443 ESTABLISHED:SYN\_SENT  (192.168.29.79:48688) 47.93.32.127:32100 MULTIPLE:SINGLE |
| | awk -F ':' '/\(192\.168\./ {print $1 "\t" $2 "\t" $3}' | (192.168.111.46 55796) 23.197.144.209 443 ESTABLISHED  (192.168.29.11 58624) 149.154.175.53 443 ESTABLISHED  (192.168.29.196 60414) 146.75.118.73 443 TIME\_WAIT  (192.168.29.3 42882) 142.251.39.106 443 ESTABLISHED  (192.168.18.197 59164) 157.240.253.13 443 ESTABLISHED  (192.168.29.27 10321) 43.157.13.47 6000 MULTIPLE |
| | awk -F ':' '/\(192\.168\./ ~~$3~/ESTABLISHED/~~ {print $1 "\t" $2 "\t" $3}' | (10.100.201.149 46278) 142.250.185.163 80 ESTABLISHED  (192.168.19.18 21427) 52.57.18.111 5222 ESTABLISHED  (192.168.29.62 37780) 142.250.179.142 443 ESTABLISHED  (192.168.18.35 38850) 108.177.15.188 5228 ESTABLISHED |
| pfctl -ss | awk '{ print $4 "\t" $6 "\t" $7 }'| awk -F ':' '/\(192\.168\./ ~~$3~/ESTABLISHED~~/ {print "TIMESTAMP" "\t" $1 "\t" $2"\t" $3}' | sed 's/[()]//g' | sort | uniq -c | TIMESTAMP (192.168.7.173 61959) 17.57.146.55 5223 ESTABLISHED  TIMESTAMP (192.168.7.173 62154) 17.57.146.27 5223 ESTABLISHED  TIMESTAMP (192.168.7.173 63210) 74.125.133.188 5228 ESTABLISHED  TIMESTAMP (192.168.7.173 63261) 157.240.253.13 443 ESTABLISHED |
|  | |
| **pfctl -ss | awk '{ print $4 "\t" $6 "\t" $7 }' | awk -F ':' '/\(192\.168\./ {print $1 "\t" $2 "\t" $3}' | sort | uniq -c >> /usr/home/ftp1c/stat168.log** | |
| **pfctl -ss | awk '{ print $4 "\t" $6 "\t" $7 }'| awk -F ':' '/\(192\.168\./ $3~/ESTABLISHED/ {print "TIMESTAMP" "\t" $1 "\t" $2"\t" $3}' | sed 's/[()]//g' | sort | uniq -c** | |
| **timestamp=$(date "+%Y-%m-%d %H:%M")**  **echo $timestamp**  **pfctl -ss | awk '{ print $4 "\t" $6 "\t" $7 }'| awk -F ':' '/\(192\.168\./ $3~/ESTABLISHED/ {print "TIMESTAMP" "\t" $1 "\t" $2"\t" $3}' | sed 's/[()]//g' | sort | uniq -c | sed "s|TIMESTAMP|$timestamp|g"** | |
| **timestamp=$(date "+%Y-%m-%d %H:%M")**  **echo $timestamp**  **pfctl -ss | awk '{ print $4 "\t" $6 "\t" $7 }'| awk -F ':' '/\(192\.168\./ $3~/ESTABLISHED/ {print "TIMESTAMP" "\t" $1 "\t" $2"\t" $3}' | sed 's/[()]//g' | sed "s|TIMESTAMP|$timestamp|g"| sort | uniq -c >> /usr/home/ftp1c/stat168.log** | |
| **pfctl -ss | awk '{ print $4 "\t" $6 "\t" $7 }'| awk -F ':' '/\(10\.10\./ $3~/ESTABLISHED/ {print "TIMESTAMP" "\t" $1 "\t" $2"\t" $3}' | sed 's/[()]//g' | sed "s|TIMESTAMP|$timestamp|g"| sort | uniq -c >> /usr/home/ftp1c/stat10.log** | |
| **cat ./99statistika.sh**  **timestamp=$(date "+%Y-%m-%d %H")**  **echo $timestamp**  **pfctl -ss | awk '$7~/ESTABLISHED/ { print $4 "\t" $6 "\t" $7 }' | awk -F ':' '/\(192\.168\./ {print $1 "\t" $2 "\t" $3}' | awk ' {print "TIMESTAMP" "\t" $1 "\t" $2"\t" $3}' | sed 's/[()]//g' | sed "s|TIMESTAMP|$timestamp|g"| sort >> /usr/home/ftp1c/stat168.log**  **pfctl -ss | awk '$7~/ESTABLISHED/ { print $4 "\t" $6 "\t" $7 }' | awk -F ':' '/\(10\.10/ {print $1 "\t" $2 "\t" $3}' | awk ' {print "TIMESTAMP" "\t" $1 "\t" $2"\t" $3}' | sed 's/[()]//g' | sed "s|TIMESTAMP|$timestamp|g"| sort >> /usr/home/ftp1c/stat10.log** | |
| **cat ./99\_4hour\_stat.sh**  **timestamp=$(date "+%Y-%m-%d %H")**  **echo $timestamp**  **cat /usr/home/ftp1c/stat10.log | sort | uniq -c > /usr/home/ftp1c/stat10\_4hour.log**  **cat /usr/home/ftp1c/stat168.log | sort | uniq -c > /usr/home/ftp1c/stat168\_4hour.log**  **rm /usr/home/ftp1c/stat10.log**  **rm /usr/home/ftp1c/stat168.logroot** | |

* 1. Обработка логов (статистики)

|  |
| --- |
| См 1С |
| Result for statistika |
| timestamp=$(date "+%Y-%m-%d %H")  echo $timestamp  pfctl -ss | awk '$7~/ESTABLISHED/ { print $4 "\t" $6 "\t" $7 }' | awk -F ':' '/\(192\.168\./  {print $1 "\t" $2 "\t" $3}' | awk ' {print "TIMESTAMP" "\t" $1 "\t" $3 "\t" $4}' | sed 's/[()]//g' | sed "s|TIMESTAMP|$timestamp|g"| sort  >> /usr/home/ftp1c/stat168.log  pfctl -ss | awk '$7~/ESTABLISHED/ { print $4 "\t" $6 "\t" $7 }' | awk -F ':' '/\(10\.10/  {print $1 "\t" $2 "\t" $3}' | awk ' {print "TIMESTAMP" "\t" $1 "\t" $3 "\t" $4}' | sed 's/[()]//g' | sed "s|TIMESTAMP|$timestamp|g"| sort  >> /usr/home/ftp1c/stat10.log |

* 1. Сохранение маков

|  |  |
| --- | --- |
|  |  |
| arp -a | awk '**\!/incomplete/**{ print $2 "\t" $4 "\t" $6 "\t" $7 }' | sed 's/[()]//g' | sort > /usr/home/ftp1c/macs.log  #arp -a | awk '\!/incomplete/{ print $2 "\t" $4 "\t" $6 "\t" $7 }' | sed 's/[()]//g' | sort > /usr/home/ftp1c/macs.log  arp -a | awk '\!/incomplete/{ print $2 "\t" $4 "\t" $6 "\t" $7 }' | sed 's/[()]//g' | sort > /usr/home/ftp1c/macs.log | |
|  |  |

# Middleware for api

|  |
| --- |
| **app.auth.plugin.js (call in** app.register(authPlugin))  const fp = require('fastify-plugin')  require('dotenv').config()  **const allowedIPAddresses = process.env.API\_ALLOWED\_IPS.split(',')**  const restrictIPMiddleware = (*req*, *reply*, *done*) => {    const clientIP = *req*.ip    if (!allowedIPAddresses.includes(clientIP)) {  *reply*.code(403).send('Forbidden')    } else {      done()    }  }  async function authPlugin(*fastify*, *\_* = {}) {  *fastify*.decorateRequest('auth', null)  ***fastify*.addHook('onRequest', restrictIPMiddleware)**  *fastify*.addHook('onRequest', async (*request*, *\_*) => {      const { authorization } = *request*.headers  *request*.auth = {        token: null,        userId: null      }      if (authorization) {        try {  *request*.auth = {            token: authorization          }        } catch (e) {          console.log(e)        }      }    })  }  module.exports = fp(authPlugin) |
|  |
| **npm install ip-range-check** |
| const fp = require('fastify-plugin')  const ipRangeCheck = require('ip-range-check')  require('dotenv').config()  const allowedSubnets = process.env.REDIRECT\_ALLOWED\_SUBNETS.split(',')  const restrictIPSubnetMiddleware = (*req*, *reply*, *done*) => {    const clientIP = *req*.ip  **if (!ipRangeCheck(clientIP, allowedSubnets)) {**      console.log(`Client IP is not allowed: ${clientIP}`)  *reply*.code(403).send('Forbidden')    } else {      console.log(`Client IP is allowed: ${clientIP}`)      done()    }  }  async function redirectPlugin(*fastify*, *\_* = {}) {  *fastify*.addHook('preHandler', restrictIPSubnetMiddleware)  }  module.exports = fp(redirectPlugin) |
| guards\is-authorized.guard.js !!!Дописать потом |
| const HttpError = require('http-errors')  module.exports = function (*request*, *\_reply*, *done*) {    if (!*request*.auth.userId) {  *//throw new HttpError.Unauthorized('Authorization required')*  *//TODO: rewrite this for this project*    }    done()  } |

# Token generation and check

npm install crypto

auth/ - route – controller – service

|  |
| --- |
| const jwt = require('jsonwebtoken')  const crypto = require('crypto')  const fs = require('fs')  const serviceAccount = JSON.parse(fs.readFileSync(process.env.GOOGLE\_APPLICATION\_CREDENTIALS, 'utf8'))  const secretKey = crypto.createHash('sha256').update(serviceAccount.private\_key).digest('hex')  module.exports.createAccessToken = async function (*payload*) {    if (process.env.ACCEPT\_CREATING\_ACCESS\_TOKENS === 'true') {  **return jwt.sign(*payload*, secretKey, { expiresIn: '1h' })**  **const expiresIn = 180 \* 24 \* 3600**  **return jwt.sign(*payload*, secretKey, { expiresIn})**    } else {      throw new Error('Access token creation is disabled')    }  }  module.exports.checkAccessToken = async function (*token*) {    try {      return jwt.verify(*token*, secretKey)    } catch (err) {      return null    }  } |
| app.auth.plugin.js |
| const fp = require('fastify-plugin')  require('dotenv').config()  const authService = require('../services/authService')  const allowedIPAddresses = process.env.API\_ALLOWED\_IPS.split(',')  const restrictIPMiddleware = (*req*, *reply*, *done*) => {    const clientIP = *req*.ip    if (!allowedIPAddresses.includes(clientIP)) {  *reply*.code(403).send('Forbidden')    } else {      done()    }  }  async function authPlugin(*fastify*, *\_* = {}) {  *fastify*.decorateRequest('auth', null)  *fastify*.addHook('onRequest', restrictIPMiddleware)  *fastify*.addHook('onRequest', async (*request*, *\_*) => {      const { authorization } = *request*.headers  *request*.auth = {        token: null,        clientId: null      }      if (authorization) {        try {          const decoded = await authService.checkAccessToken(authorization)  *request*.auth = {            token: authorization,            clientId: decoded.clientId          }        } catch (e) {          console.log(e)        }      }    })  }  module.exports = fp(authPlugin) |
| is-authorized.guard.js |
| const HttpError = require('http-errors')  module.exports = function (*request*, *\_reply*, *done*) {    if (!*request*.url.includes('/redirect')) {      if (!*request*.auth.clientId) {        throw new HttpError.Unauthorized('Authorization required')      }    }    done()  } |

# Файл .env & CURL

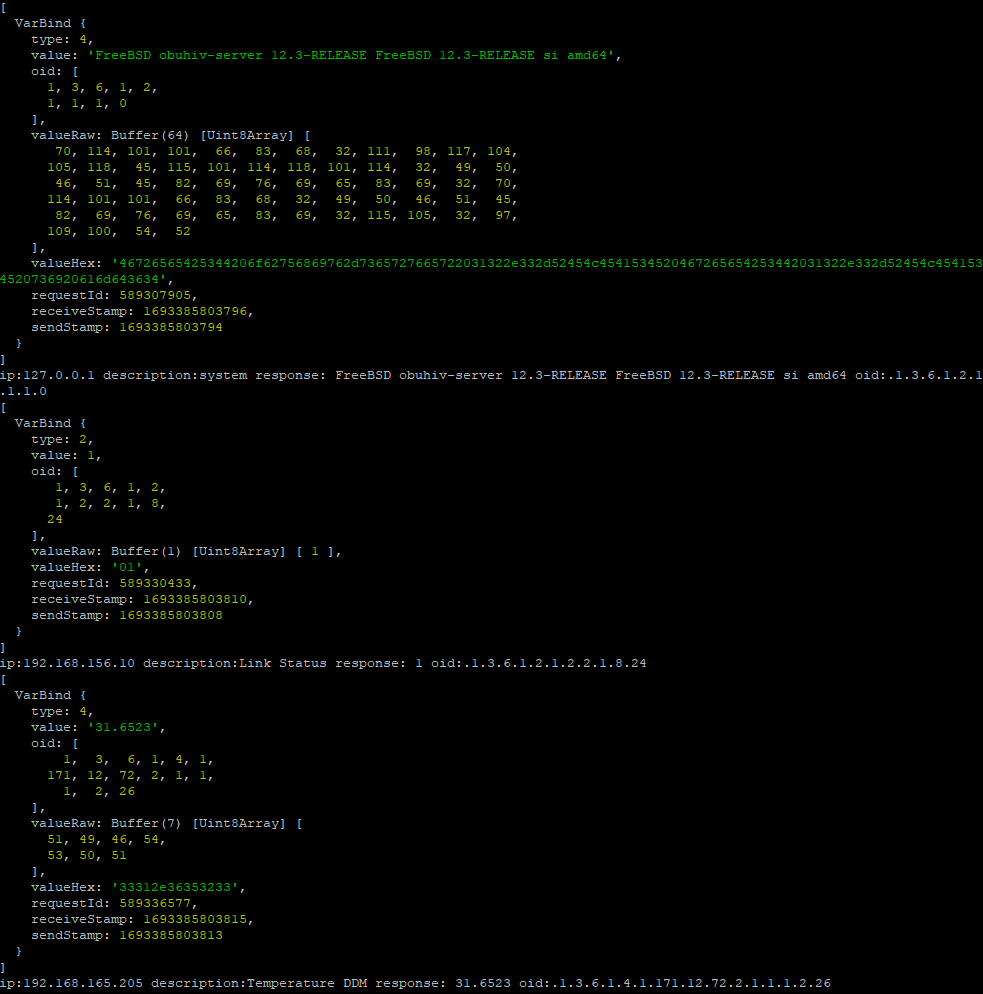
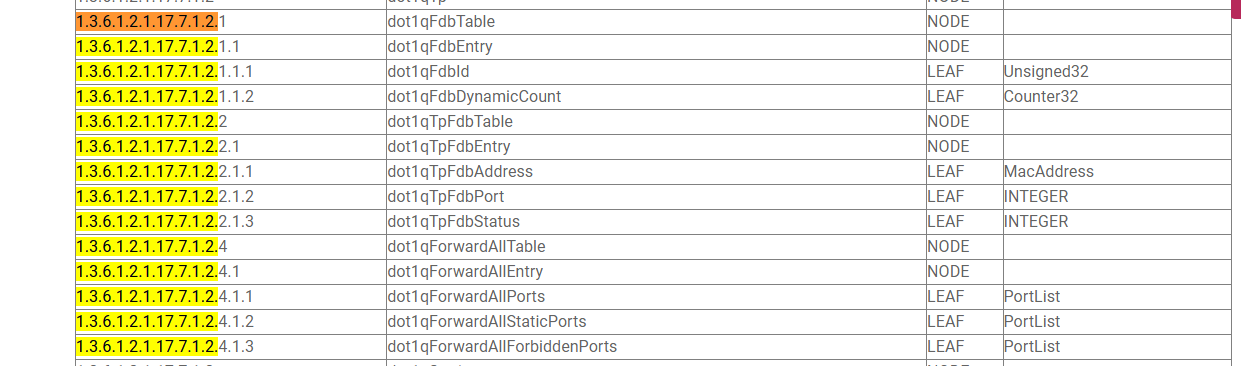
|  |
| --- |
| PORT = 8000  PORT\_FOR\_REDIRECT = 8001  UPSTREAM\_URL = http://www.silver-service.com.ua/redirect.html  PLATFORM = windows  PF\_LOG\_FILE168 = g:\10Silvers\2023\FreeBSD\StstisticsFormat\stat168\_4hour.log  PF\_LOG\_FILE10  = g:\10Silvers\2023\FreeBSD\StstisticsFormat\stat10\_4hour.log  MACS\_FILE = g:\10Silvers\2023\FreeBSD\StstisticsFormat\macs.log  SERVER\_ID = SW2  FIRESTORE\_DB\_URL = https://your-firestore-db-url.firebaseio.com  GOOGLE\_APPLICATION\_CREDENTIALS = "./path/to/serviceAccountKey.json"  SAVE\_TO\_FIRESTORE = true  SAVE\_TO\_LOCAL\_DB  = true  ACCEPT\_REMOVE\_COLLECTION = false  ACCEPT\_CREATING\_ACCESS\_TOKENS = true  TOKER\_EXPIRE\_IN\_DAYS = 180  URL = http://127.0.0.1:80/billing/hs/query/Request  AUTH\_TOKEN = Basic ххххххххххххх  **API\_ALLOWED\_IPS** = 127.0.0.1,::1,91.220.106.11  **REDIRECT\_ALLOWED\_SUBNETS** = 127.0.0.1,192.168/16,10/8,172.16/12 |
| Пример вызова метода **\api** из crontab на UNIX |
| url="http://127.0.0.1:8000/api/trafficAnalyze/log-saving/"  body='{    "srcIpAddress": "\*",    "dstIpAddress": "\*",    "startDate": "2023-07-20T12:00:00",    "endDate": "2023-07-20T12:59:59"  }'  *# Set the authorization header value*  authorization='insert the token'  *# Make the POST request using curl*  curl -X POST "$url" \       -H "Authorization: $authorization" \       -H "Content-Type: application/json" \       -d "$body" |

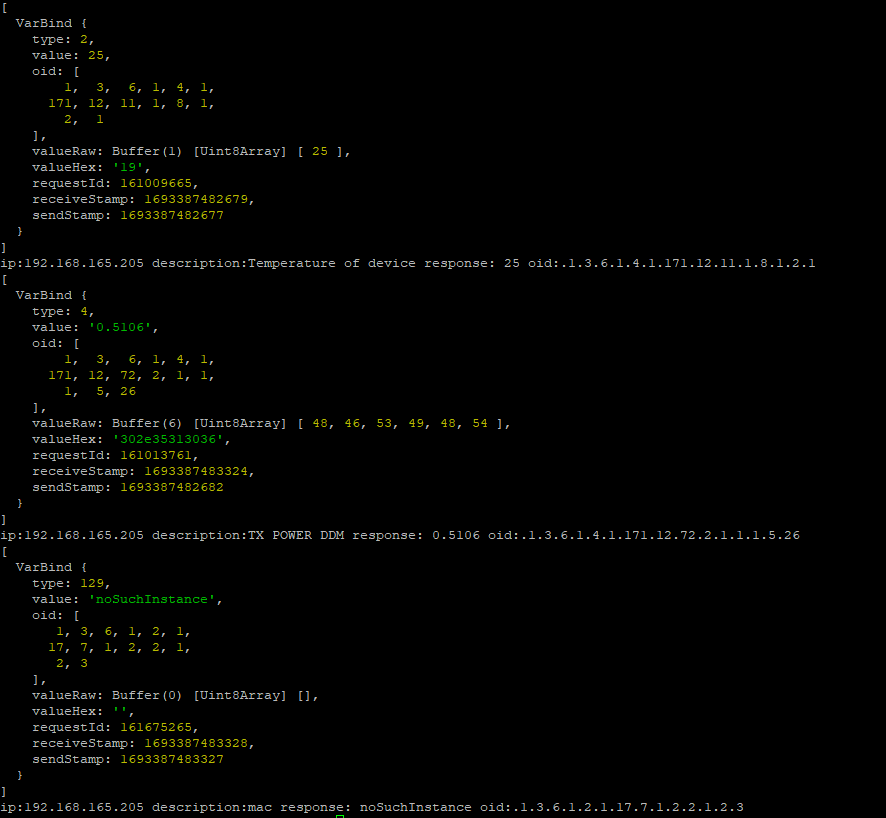
# Launching on FreeBSD.

|  |
| --- |
| [~~https://www.openssl.org/source/~~](https://www.openssl.org/source/)  ~~tar -xzvf openssl-1.1.1u.tar.gz~~  ~~./config~~  ~~Make~~  ~~Make install~~ |
|  |

# Snmp and debug on FreeBSD.

|  |  |
| --- | --- |
| snmp-tests,js | |
| const snmp = require('snmp-native')  const testSnmpObjectsList = [    { ip\_address: '192.168.0.1', description: 's1', oid: '.1.3.6.1.2.1.1.1.0', status: 'dead' },    { ip\_address: '192.168.165.205', description: 's2', oid: '.1.3.6.1.2.1.2.2.1.8.2', status: 'dead' },  ]  async function snmpGet(*snmpObject*, *community* = 'public') {    const session = new snmp.Session({ host: *snmpObject*.ip\_address, community: *community*, timeout: 5000 })    try {      const varbinds = await new Promise((*resolve*, *reject*) => {        session.get({ oid: *snmpObject*.oid }, (*error*, *varbinds*) => {          session.close()          if (*error*) {            reject(*error*)          } else {            resolve(*varbinds*)          }        })      })      if (varbinds.length > 0) {        return varbinds[0].value      } else {        throw new Error('No response received')      }    } catch (error) {      console.error('Error:', error) *// Log the error details*      throw error    }  }  async function main() {    for (const snmpObject of testSnmpObjectsList) {      try {        const response = await snmpGet(snmpObject)        console.log(response)      } catch (error) {        console.error(`Error querying ${snmpObject.ip\_address}:`, error)      }    }  }  main() | |
| **npm install snmp-native**  **node ./snmp-tests.js** | |
| **1.3.6.1.2.1.2.2.1.8.24** | is the OID for the "**ifOperStatus**" object, and **24** is the interface(port) index.   * 1: Up * 2: Down * 3: Testing * 4: Unknown * 5: Dormant * 6: NotPresent * 7: LowerLayerDown |
| **.1.3.6.1.4.1.171.12.72.2.1.1.1.2.26** | **Temperatura DDM port 26 – the answer is 31.5039** |
| **.1.3.6.1.4.1.171.12.11.1.8.1.2.1** | **Temp status** |
|  | **.1.3.6.1.2.1.17.7.1.2**  **.1.3.6.1.2.1.17.7.1.2.2.1.2** |
| **VlanXX** | **.1.3.6.1.2.1.17.7.1.2.2.1.2.XX** |
| **snmpwalk -v 2c -c public -OXsq -On 127.0.0.1 .1.3.6.1.2.1.17.7.1.2.2.1.2.XX**  **snmpwalk -v 2c -c public -OXsq -On 127.0.0.1 .1.3.6.1.2.1.17.7.1.2.2.1.2.XXX** | |
| const testSnmpObjectsList = [    { ip\_address: '127.0.0.1', description: 'Port24 220V Link Status', oid: '.1.3.6.1.2.1.2.2.1.8.24', min: '', max: '', value: '1', status: 'dead' },    { ip\_address: '127.0.0.1', description: 'Temperature Status Switch ', oid: '.1.3.6.1.4.1.171.12.11.1.8.1.2.1', min: '18', max: '41', value: '', status: 'dead' },    { ip\_address: '127.0.0.1', description: 'DDM Temperature Status Port25', oid: '.1.3.6.1.4.1.171.12.72.2.1.1.1.2.25', min: '15', max: '67', value: '', status: 'dead' },    { ip\_address: '127.0.0.1', description: 'Rx Power DDM\_25\_UP', oid: '.1.3.6.1.4.1.171.12.72.2.1.1.1.6.25', min: '-12', max: '-11.5', value: '', status: 'dead' },    { ip\_address: '127.0.0.1', description: 'Tx Power DDM\_25\_UP', oid: '.1.3.6.1.4.1.171.12.72.2.1.1.1.5.25', min: '2.3', max: '2.9', value: '1', status: 'dead' },  ]  const snmp = require('snmp-native') | |
| async function snmpGet(*snmpObject*, *community* = 'public') {    const session = new snmp.Session({ host: *snmpObject*.ip\_address, community: *community*, timeout: 5000 })    try {      const varbinds = await new Promise((*resolve*, *reject*) => {        session.get({ oid: *snmpObject*.oid }, (*error*, *varbinds*) => {          session.close()          if (*error*) {            reject(*error*)          } else {            resolve(*varbinds*)          }        })      })      if (varbinds.length > 0) {        return snmpAnswersAnalizer(*snmpObject*, varbinds)      } else {        throw new Error('No response received')      }    } catch (error) {      console.error('Error:', error)      throw error    }  }  function snmpAnswersAnalizer(*snmpObject*, *varbinds*) {    try {      if (*varbinds*[0].type === 2 && *snmpObject*.value !== '') {        if (*varbinds*[0].value === Number(*snmpObject*.value)) {          return 'Status OK'        }      }      if (*varbinds*[0].type >= 2 && (*snmpObject*.min !== '' || *snmpObject*.max !== '')) {        if (*varbinds*[0].value >= Number(*snmpObject*.min) && *varbinds*[0].value <= Number(*snmpObject*.max)) {          return `value ${*varbinds*[0].value} Status OK`        } else {          return `value ${*varbinds*[0].value} Status PROBLEM`        }      }      return *varbinds*[0].value    } catch (error) {      console.error('Error:', error)      throw error    }  } | |
|  | |





# Решение проблемы асинхронности child\_process

|  |
| --- |
| const util = require('util');  const exec = util.promisify(require('child\_process').exec);  async function snmpwalk(args, value) {  const command = `snmpwalk ${args.join(' ')}`;    try {  const { stdout, stderr } = await exec(command);  if (stdout.includes(value)) {  console.log('+++++++++');  return 'Status OK';  } else {  console.log('------');  return 'Status NOT OK';  }  } catch (error) {  console.error(`Ошибка выполнения команды: ${error.message}`);  return 'Ошибка';  }  }  async function main() {  const response = await snmpwalk(['-v', '2c', '-c', 'public', '-OXsq', '-On', '192.168.156.10', '.1.3.6.1.2.1.17.7.1.2.2.1.2.616'], '164.147.76.110.114.120');  console.log('response', response);  }  main(); |

# Apache & redirect + ipfw fwd

|  |
| --- |
| Listen XXX , where XXX is a basical port  **Listen XXX1** , where XXX1 is a additional port |
| **<VirtualHost \*:7878>**  **DocumentRoot "${SRVROOT}/htdocs"**  **ServerName 127.0.0.1**  **Redirect / http://127.0.0.1:80/redirect.html**  **</VirtualHost>** |
|  |

# switch-redir + fwd-off

|  |
| --- |
| '/abonents/switch-redir/' (abonentsService.switchRedir)  POST[**http://127.0.0.1:8000/api/abonents/switch-redir/**](http://127.0.0.1:8000/api/abonents/switch-redir/) **CALL FROM BILLING =**= DEBUG + |
| const filePath = `/home/admin/deny\_ip/vlan${vlanId}\_deny\_hosts`      const content = ipAddresses.map(ip => ip + '\n').join('')      await writeFile(filePath, content)      console.log(`${new Date()}: Redir file for vlan=${vlanId} wrote successfully.`)      const addCommand = await runCommand('/sbin/pfctl -nf /etc/pf.rules')      if (addCommand.stdout) {        console.log(`${new Date()}: pf rules uploaded for vlan=${vlanId} successfully.`)      } |
| GET **http://127.0.0.1:8002/redirect-api/service-go-on/ CALL FROM www =**= DEBUG + |
| module.exports.execServiceContinued = async function (ipAddress) {    if (process.env.PLATFORM !== 'freebsd') {      return true    }    try {      const parts = ipAddress.split('.')      const vlanId = parts.length === 4 ? parts[2] : null      if (!vlanId) {        console.error(`Invalid ipAddress=${ipAddress} format. Could not extract vlanId.`)        return false      }      const filePath = `/home/admin/deny\_ip/vlan${vlanId}\_deny\_hosts`      if (!locks[filePath]) {        locks[filePath] = Promise.resolve();      }      await locks[filePath]      const fileContent = await readFile(filePath, 'utf8')      const updatedContent = fileContent        .split('\n')        .filter(line => line.trim() !== ipAddress)        .join('\n')      await writeFile(filePath, updatedContent)      console.log(`${new Date()}: Removed ${ipAddress} for vlan=${vlanId} successfully.`)      locks[filePath] = Promise.resolve()      const addCommand = await runCommand(`/sbin/pfctl -nf /etc/pf.rules`)      if (addCommand.stdout) {        console.log(`${new Date()}: pf rules uploaded for vlan=${vlanId} successfully.`)      }      return true    } catch (error) {      console.error('Error executing commands:', error.message)      return false    }  } |
|  |