Waverley Labs

Open Source Software Defined Perimeter

Installation and Configuration

Version 0.2

July 2020

Table of Contents

[Supported Platforms 3](#_Toc474851739)

[SDP Controller 3](#_Toc474851740)

[SDP Gateway 6](#_Toc474851741)

[SDP Client 9](#_Toc474851742)

# Supported Platforms

The SDP Client is currently only being tested on macOS along with Debian and RHEL versions of Linux. It is unlikely to function on Windows at present. Support for other platforms will be provided in the future.

The SDP Gateway is currently only being tested on Debian and RHEL versions of Linux. It is unlikely to function on other operating systems at present. Support for other platforms is TBD.

The SDP Controller is currently only being tested on Debian and RHEL versions of Linux. Because it runs in a node.js framework, it may function on macOS and Windows, but this is not guaranteed.

# SDP Controller

\*\*NOTE\*\* The SDP Controller is currently only being tested on Debian and RHEL versions of Linux. Because it runs in a node.js framework, it may function on macOS and Windows, but this is not guaranteed.

1. Install Node.js on your system. For details, see <https://nodejs.org/en/download/>
2. Install the node package manager (npm) if it was not automatically installed in step 1. Check by opening a terminal and entering:   
     
   npm
3. Download or clone the SDP Controller open source project from GitHub at:  
     
   <https://github.com/WaverleyLabs/SDPcontroller>
4. Install the node packages that this project requires. Do this by navigating to the source code project folder in a terminal and entering:   
     
   npm install
5. Install MySQL.
6. In MySQL, import the sample database provided with this project in file *./setup/sdp.sql*
7. In MySQL, setup a user with write privileges for this new database.
8. In MySQL, populate the relevant tables with controller, gateway, and client information. At a minimum, a basic setup would have at least one controller, gateway, and client, requiring the following tables to be populated:
   1. sdpid – at least one entry for each of the three required components
   2. service – the controller should be included as an entry in this table
   3. service\_gateway – for each active service, there should be at least one entry declaring the gateway(s) by which it is protected. There can be multiple instances of a service and/or multiple gateways protecting a service instance.
   4. sdpid\_service – which SDP IDs have access to a service. Remote gateways must have an entry here, providing access to the controller if the controller is behind another gateway.

The controller can also use ‘groups’ to provide access to services. In this scenario, rather than adding an entry for a SDP ID in the sdpid\_service table above, the administrator can create entries in the following additional tables:

1. user
2. group
3. user\_group
4. group\_service

The following tables are currently placeholders that may prove important later:

1. controller
2. gateway
3. gateway\_controller
4. In the SDPcontroller project, edit *./config.js* based on previous steps. The options are explained throughout the configuration file.
5. If not already installed, install openssl
6. Create a certificate authority key and certificate using the following commands in the terminal:  
     
   openssl genrsa -des3 -out ca.key 4096  
   openssl req -new -x509 -days 365 -key ca.key -out ca.crt
7. Generate sample keys and certs for each SDP component. This step assumes that a complete set of information about each SDP ID (the controller, gateway(s), and client(s)) was entered into the database. In a terminal, navigate to the SDP controller source code directory and execute:  
     
   node ./genCredentials.js SDPID  
     
   where SDPID is an existing ID in the database such as 12345. This command will create three new files in the current directory: *12345.crt*, *12345.key*, and *12345.spa\_keys*. The program will store the SPA keys in the controller database automatically. The *crt* and *key* files must be transferred to the device in question. The *spa\_keys* file is only for added convenience in copying the SPA key material to the intended device. The file itself is not used by any of the components. The SPA key material must be entered into the respective device’s *.fwknoprc* file and *sdp\_ctrl\_client.conf* file.
8. To start the controller, in a terminal enter:   
     
   node ./sdpController.js

# SDP Gateway

\*\*NOTE\*\* The SDP Gateway is currently only being tested on Debian and RHEL versions of Linux. It is unlikely to function on other operating systems at present. Support for other platforms is TBD.

1. Ensure that iptables is installed and operating on the gateway machine.
2. Configure iptables (this assumes a fairly deep understanding of iptables):
   1. Establish drop-all policies for all traffic
   2. Make sure that connection tracking is enabled
   3. Ensure that traffic for established connections is permitted using rules like the following:  
        
      iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT  
        
      The same type of rule should be used for forwarding and outbound traffic if applicable.
   4. Enable forwarding if applicable using the following:  
        
      echo 1 > /proc/sys/net/ipv4/ip\_forward
3. Install libraries that fwknop depends on to build using the following command in a terminal:  
     
   sudo apt-get install net-tools conntrack openssl libssl-dev libjson-c-dev libpcap-dev texinfo libtool autoconf libuv1 libuv1-dev
4. Download or clone the SDP Gateway/Client open source project (a fork of the ‘fwknop’ project) from GitHub at:  
     
   <https://github.com/waverleylabs/fwknop>
5. In a terminal, navigate into the top-level directory of the fwknop project.
6. Enter the following commands to prepare the project for building:  
     
   libtoolize --force  
   aclocal  
   autoheader  
   automake --force-missing --add-missing  
   autoconf
7. Configure the project for compiling by entering:  
     
   ./configure --prefix=/usr --sysconfdir=/etc --with-iptables=/path/to/iptables  
     
   \*\*NOTE\*\* The command above configures the project to build both client and gateway programs as well as install them. You can add one of the following options to the command above in order to limit the build and install to only one of the two components:  
     
   --disable-server  
   --disable-client
8. Compile by entering:  
     
   make
9. Install by entering:  
     
   sudo make install  
     
   \*\*NOTE\*\* If there are errors with any of these steps, it may help to perform the following steps in the terminal and then repeat the steps above:  
     
   libtoolize --force  
   aclocal  
   autoheader  
   automake --force-missing --add-missing  
   autoconf
10. Edit the following configuration files, which should now be located in /etc/fwknop/ unless different options were passed in the configure step above. Each has explanations throughout:
    1. *fwknopd.conf*
    2. *gate\_sdp\_ctrl\_client.conf* – contains information for connecting to a local or remote controller.
    3. *gate.fwknoprc* – only required on remote gateways that must communicate with the controller *through* the controller’s local gateway. Remote gateways must initially send SPA like any other client so that the controller’s local gateway opens a path for the remote gateway. In order to send the SPA packet, the gateway calls the fwknop client program, using the settings in this file to do so.
11. Start the gateway by entering:  
      
    fwknopd  
      
    If the controller is running and accessible, the gateway should connect and retrieve all necessary data to operate.

# SDP Client

\*\*NOTE\*\* The SDP Client is currently only being tested on macOS along with Debian and RHEL versions of Linux. It is unlikely to function on Windows at present. Support for other platforms will be provided in the future.

1. On the client device, download or clone the SDP Gateway/Client open source project (a fork of the ‘fwknop’ project) from GitHub at:  
     
   <https://github.com/waverleylabs/fwknop>
2. In a terminal, navigate into the top-level directory of the fwknop project.
3. Configure the project for compiling by entering:  
     
   **LINUX**  
   sudo apt-get install net-tools openssl libssl-dev libjson-c-dev libpcap-dev texinfo libtool autoconf make telnet libuv1 libuv1-dev  
     
   libtoolize --force  
   aclocal  
   autoheader  
   automake --force-missing --add-missing  
   autoconf  
   ./configure --disable-server --prefix=/usr  
     
   **MAC**brew install texinfo  
   brew install libtool  
   brew install autoconf  
   brew link --overwrite autoconf  
   brew install make  
   add to .bashrc file: PATH="/usr/local/opt/make/libexec/gnubin:$PATH"  
   brew install openssl (this includes dev too)  
   ln -s /usr/local/opt/openssl/include/openssl /usr/local/include  
   ln -s /usr/local/opt/openssl/lib/lib\* /usr/local/lib/  
   brew install json-c  
   brew install libuv  
   glibtoolize --force  
   aclocal  
   autoheader  
   automake --force-missing --add-missing  
   autoconf./configure --disable-server  
   *\*NOTE\* - for configure step above, using –prefix=/usr on Mac caused link warnings and errors*
4. Compile by entering:  
     
   make
5. Install by entering:  
     
   sudo make install  
     
   \*\*NOTE\*\* If there are errors with any of these steps, it may help to perform the following steps in the terminal and then repeat the steps above:  
     
   libtoolize --force  
   aclocal  
   autoheader  
   automake --force-missing --add-missing  
   autoconf
6. Check that fwknop was successfully installed by entering:  
     
   fwknop -V
7. In the fwknop project, in the client folder, copy the files *SAMPLE.fwknoprc* and *SAMPLE\_sdp\_ctrl\_client.conf* to the user’s home directory. Rename *SAMPLE.fwknoprc* to *.fwknoprc*.
8. Open *.fwknoprc* for editing (eventually this will all be automated)
   1. This file is broken down into “stanzas.” Each stanza has a name which is enclosed in brackets. Typically, when calling the fwknop client, the user passes a stanza name to the client. Each stanza tells the client how to generate and send SPA to gain access to a particular service or group of services (if you wanted to access multiple services simultaneously). Any variables set under the [default] stanza are applied to all stanzas unless the chosen stanza overrides a variable.
   2. In this sample *.fwknoprc* file, there is a stanza named [sdp\_ctrl\_gate]. This stanza is used by the client to gain access through the controller’s gateway. Configure the variables as follows:
      1. ALLOW\_IP: The public address of the client to permit through the firewall. The entry can be one of:
         1. the device’s public IP address
         2. resolve - the client program will request the host’s public IP from the site <https://www.cipherdyne.org/cgi-bin/myip> by default though this is configurable as well
         3. source - ask the gateway to use the source address of the SPA packet’s IP header. This is the least secure method.
      2. SPA\_SERVER: The address of the gateway protecting the controller or other service.
      3. The other variables should be self-explanatory. See this website for additional information on all possible variables:   
           
         <http://www.cipherdyne.org/fwknop/docs/manpages/fwknop.html>
   3. There is another stanza in *.fwknoprc* named [service\_gate]. This is just another example stanza for the client to gain access through a gateway to a service. Notice the additional variable SDP\_CTRL\_CLIENT\_CONF. Whenever a service has this variable set, by default, the SDP client program will first send a SPA packet to the intended service gateway to enable access to the desired service. The program will then proceed to use the configuration file indicated by this variable to connect to the SDP controller, first using the [sdp\_ctrl\_gate] stanza to send SPA to the controller gateway and then other settings in the configuration file to complete the connection. Edit and duplicate this stanza as appropriate.
9. Edit *SAMPLE\_sdp\_ctrl\_client.conf* as appropriate. Detailed explanation is provided throughout the file.
10. Run the fwknop client to gain access to a protected service as well as receive credential updates from the controller by entering the following command in a terminal:  
      
    fwknop -n service\_gate