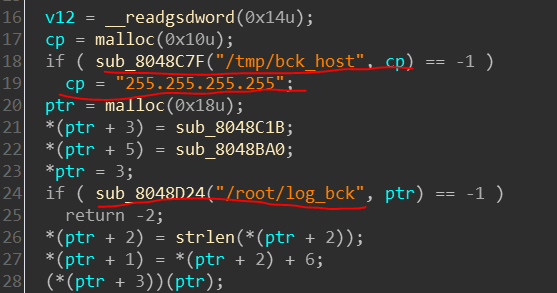
**Решение заданий категории Pwn.**

**Задание №4. «Root backdoor»**

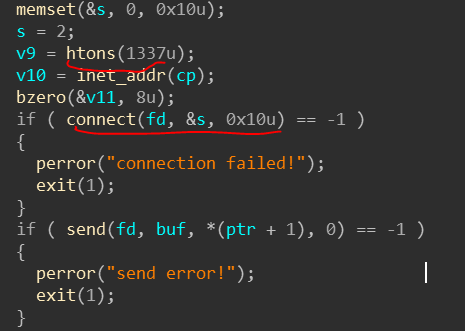
После предыдущего задания мы получили оболочку на хосте. Можно сделать её поудобнее, например, через netcat открыть порт и при подключении давать оболочку или что-то другое (вариантов много). Есть вариант сделать ключ для подключения по ssh и положить его в папку, чтобы после 3 задания можно было подключиться по ssh из-под обычного юзера.

Так или иначе нам надо повысить привилегии до root’a. Поискав файлы в домашней директории можно увидеть исполняемый файл backdoor.

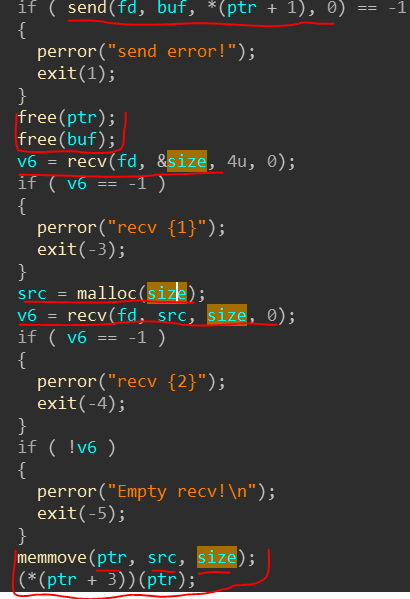
Начальный анализ позволяет понять, что данный файл является сетевым клиентом и отправляет некоторые данные на удалённый сервер, при это IP-адрес сервера считывается из файла в папке /tmp. Данные, которые отправляются на сервер, считываются из файла в директории /root.



Попробуем добавить свой IP-адрес и взглянем на трафик. Раз в минуту будут видны попытки подключения к TCP-порту 1337. В коде также можно это увидеть.



Посмотрим повнимательнее на код, после удачного подключения.

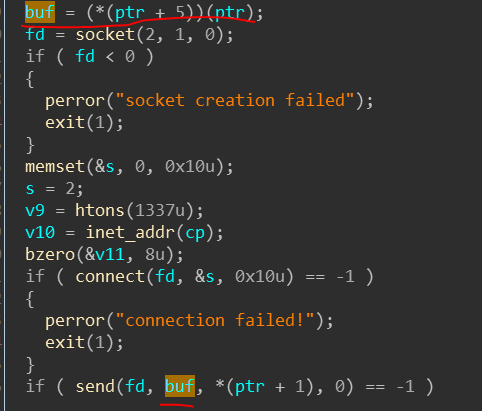


После подключения идёт отправка некоторого буфера (позже вернёмся к нему). Далее идёт освобождения память для двух указателей, один из них наш буфер, а другой – некоторая структура. После чего происходит считывание данных в переменную size, которая отвечает за размер выделяемой памяти для следующего пакета. То есть сервер отправляет 4 байта, в которых указывает размер следующего пакета. Выделяется память и туда считывается следующий пакет.

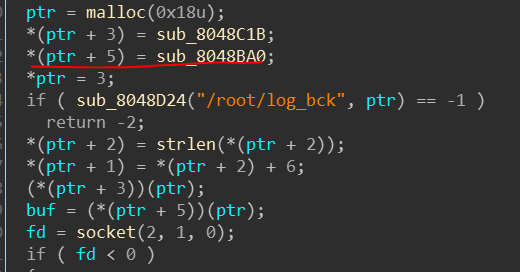
После чего происходит копирование пришедшего пакета на освобождённую память, где находилась некоторая структура и следующая команда вызывает функцию по указателю, хранящемуся в этой структуре. Видно, что можно контролировать значение данного указателя и вызвать произвольную функцию, однако мы имеем дело с сетевым кодом и для получения какой-либо оболочки нужно выполнение кода и, например, reverse\_tcp shell.

Также можно заметить, что NX отключен, значит можно выполнять код на стеке и куче.

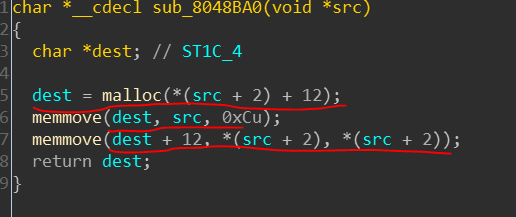
Внимательно изучим приходящие данные на сервер.



Буфер формируется вызовом некоторый функции, указатель которой хранится в структуре.



Просмотрим данную функцию. Она достаточно проста, выделяется память исходя из значения внутри структуры + 12 байт. Первые 12 байт являются непосредственно значениями структуры, а последующие копируются из некоторого указателя, содержащегося в структуре.

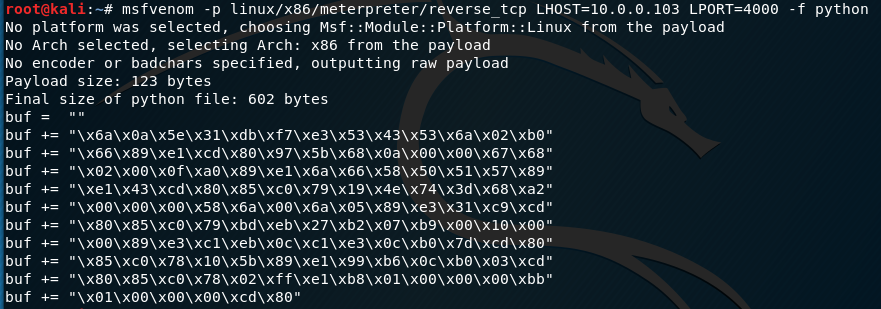


Если немного разверёстить функцию чтения лог-файла, то понятно, что данные лога сохраняются в динамическую память и указатель помещается в структуру. В целом данная структура представляет собой некоторый формат пакета, а также содержит «методы» работы с данными пакета.

Но, интересно другое, а именно то, что указатель на данные лога, которые расположены на куче, также попадает в финальный пакет и отправляется серверу, таким образом мы можем примерно предугадать адрес на куче, куда будет помещён наш пакет. С помощью отладчика можно внимательно изучить поведение кучи и примерно прикинуть смещения + сильно может помочь большая NOP-цепочка перед шеллкодом.

Остаётся написать сервер, правильно распарсить приходящий пакет, примерно подсчитать адрес, составить пейлоад с сгенерированным заранее шелкодом.

Шелкод можно сгенерировать с помощью msfvenom.



После в msfconsole запустить перехватчик и ожидать подключения.

Примерный код сервера, который обрабатывает запрос и отправляет пейлоад представлен в файле «remote\_server.py». После чего ждать, т.к. коннект идёт раз в минуту.

