Подробное описание решения задания «Arbacoin wallet» с Кубка CTF 2019.

Категория: Reverse

Ориентировочная сложность: Средняя (приблизительно 1-2 часа на решение).

Легенда: «По нашим оперативным данными, верхушка корпорации Arbalest of Siberia использует защищённые криптокошельки. Силами группировок и жизнью несколько добропорядочных агентов. Но оно запрашивает пароль на каждый кошелёк. Возможно у тебя получится разобраться с этим. Мы также нашли сервер, где генерируются эти кошельки. Говорят, что в них есть некоторая приватная информация. Добудь её и сопротивление щедро отблагодарит тебя.»

Решение.

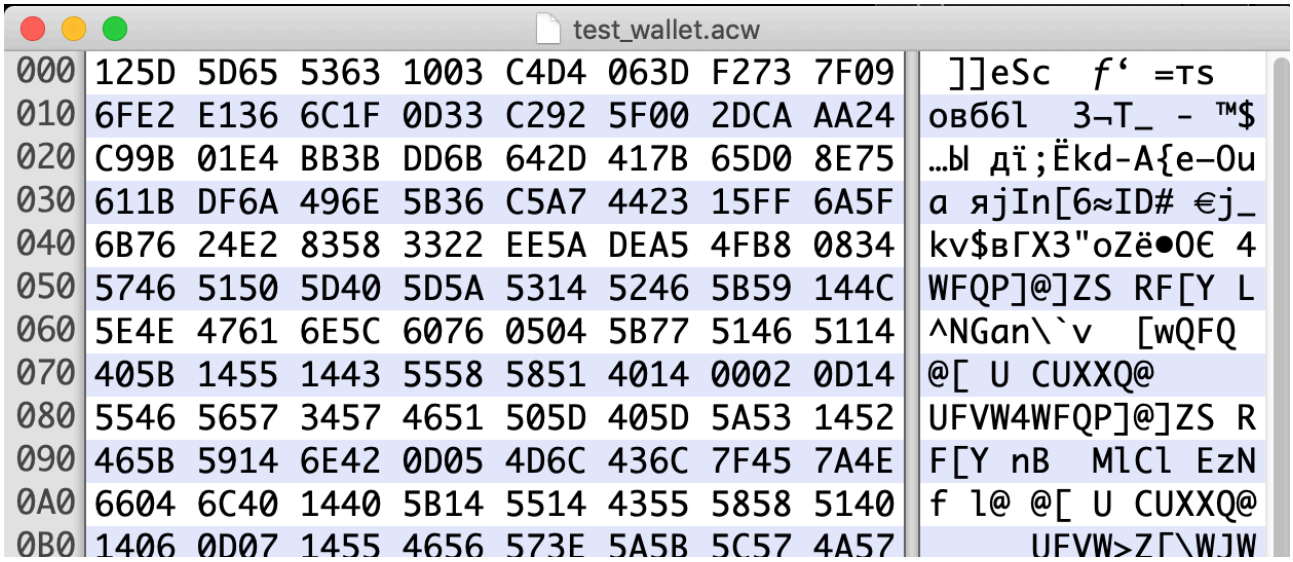
Команда получает исполняемый файл под Linux x64, написанный на C++ без методов антиотладки, с символьной информацией и ip-адрес и порт сервера, генерирующего кошельки.

1. Анализ сервера и анализ формата файла

При подключении к серверу он отдаёт некоторый блок данных в 16-ричном формате и закрывает соединение (изображение ниже).

```
➜ solve nc 10.0.0.106 7171
10714a28662966714a577e57436256214910555446 7f565877625e28667c78226357deafbb47162246 4
4a47024341414d574c560210161102435040411e7d6c7b7a776a7770793e6a713e7f3e697f72727b6a3
776a7770793e6a713e7f3e697f72727b6a3e2f27273e7f6c7c7d224647404b564b4c450244504d4f025
1002435040412246474 04b564b4c450244504d4f02564a47024341414d574c560211111b0243504041 2
02564a47024341414d574c560211161702435040411e7d6c7b7a776a7770793e6a713e7f3e697f72727
404b564b4c450244504d4f02564a47024341414d574c560213121b0243504041224647404b564b4c450
574c560213121402435040411e7d6c7b7a776a7770793e6a713e7f3e697f72727b6a3e2c2f2f3e7f6c7
4d4f02564a47024341414d574c560216111b0243504041224647404b564b4c450244504d4f02564a470
5040411e7d6c7b7a776a7770793e6a713e7f3e697f72727b6a3e2a2b2e3e7f6c7c7d224647404b564b4
414d574c560210101302435040411e7d6c7b7a776a7770793e6a713e7f3e697f72727b6a3e2d292c3e7
3e6a713e7f3e697f72727b6a3e2d28263e7f6c7c7d224647404b564b4c450244504d4f02564a4702434
41224647404b564b4c450244504d4f02564a47024341414d574c560210131702435040412246474 04b5
4341414d574c5602111b1a02435040411e7d6c7b7a776a7770793e6a713e7f3e697f72727b6a3e2f2b2
450244504d4f02564a47024341414d574c5602131a1602435040411e7d6c7b7a776a7770793e6a713e7
6c7c7d45222f480d00465039260f7363477108387573330a737156407b0942090d501221546f0a400e4
5c752e03432d5a4e540d57167e506f591344
➜ solve nc 10.0.0.106 7171
106a7f795241756061472365774071642610285a2344756362782843512854667849dea60fdc131e7d6
697f72727b6a3e2d2b2c3e7f6c7c7d224647404b564b4c450244504d4f02564a47024341414d574c560
564b4c450244504d4f02564a47024341414d574c560211161502435040411e7d6c7b7a776a7770793e6
2e3e7f6c7c7d1e7d6c7b7a776a7770793e6a713e7f3e697f72727b6a3e2c2b2c3e7f6c7c7d1e7d6c7b7
72727b6a3e2c2f2f3e7f6c7c7d224647404b564b4c450244504d4f02564a47024341414d574c5602101
7770793e6a713e7f3e697f72727b6a3e2d2b273e7f6c7c7d224647404b564b4c450244504d4f02564a4
4350404122464740 4b564b4c450244504d4f02564a47024341414d574c560211141b02435040411e7d6
697f72727b6a3e2c2a2a3e7f6c7c7d224647404b564b4c450244504d4f02564a47024341414d574c560
564b4c450244504d4f02564a47024341414d574c560216151302435040411e7d6c7b7a776a7770793e6
2e3e7f6c7c7d224647404b564b4c450244504d4f02564a47024341414d574c560211141002435040412
02564a47024341414d574c560210101102435040412246474 04b564b4c450244504d4f02564a4702434
411e7d6c7b7a776a7770793e6a713e7f3e697f72727b6a3e2c2c2f3e7f6c7c7d1e7d6c7b7a776a77707
2c2c263e7f6c7c7d45391a193968554112365241576556 4d57087d5631504a435a0d62755d2114093f4
4d010928016756151659 5b3179002113 0d604b5a08272c
➜ solve
```

Несложно догадаться, что это кошельки в 16-ричном виде и нужно преобразовать их в адекватный вид. Для этого достаточно написать небольшой скрипт, который подключается к серверу и декодирует получаемый блок данных. На самом деле достаточно одного кошелька, потому что флаг содержится в каждом. Можно просто взять полученный 16-ричный буфер и вставить в любой hex-редактор, после сохранить и получить готовый кошелёк. Кошелёк в 16-ричном редакторе выглядит приблизительно так (изображение ниже).



До непосредственного анализа исполняемого файла, можно было провести небольшое исследование устройства получаемого кошелька. Если получить ещё несколько кошельков, то можно проследить, что первое значение изменяется в небольшом диапазоне и скорей всего описывает размер последующих данных. Следующий блок выглядит аналогичным образом. Поэтому можно сделать предположение о том, что блоки сохраняются в формате <size><data>.

2. Исследование исполняемого файла.

Исполняемый файл представляет собой некоторую «читалку» кошельков. Он принимает на вход первым аргументом путь до кошелька, а после запрашивает логин и пароль. Судя по всему они сохраняются в кошельке. Посмотрим код запроса логина и пароля (изображение ниже).

Можем заметить два метода «SetUsername» и «SetPassword» которые инициализируют свойства объекта Wallet. С помощью этих функций можно узнать смещения по которому сохраняются указатели на логин и пароль (это пригодится далее).

```
Wallet::Wallet(&WalletObj, &v30);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v30);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&Login);
std::operator<<<std::char_traits<char>>(&std::cout, "Enter login: ");
std::operator>><char,std::char_traits<char>,std::allocator<char>>(&std::cin, &Login);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&Password);
std::operator<<<std::char_traits<char>>(&std::cout, "Enter password: ");
std::operator>><char,std::char_traits<char>,std::allocator<char>>(&std::cin, &Password);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&wallet_Username, &Login
v5 = Wallet::SetUsername(&WalletObj, &wallet_Username) ^ 1;
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&wallet_Username);
if ( v5 )
{
  v6 = std::operator<<<std::char_traits<char>>(&std::cout, "[-] Some error in login set!");
  std::ostream::operator<<(v6, &std::endl<char,std::char_traits<char>>);
  v4 = 259;
}
else
{
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(
    &wallet_Password,
    &Password);
  v7 = Wallet::SetPassword(&WalletObj, &wallet_Password) ^ 1;
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&wallet_Password);
  if ( v7 )
```

Анализируя код далее можно понять, что меню по просмотру кошелька будет выведено, если он будет успешно расшифрован.

```
else
{
  if ( !(Wallet::Decrypt(&WalletObj) ^ 1) )
  {
    while ( 1 )
    {
      v10 = std::operator<<<std::char_traits<char>>(&std::cout, "-- Menu --");
      std::ostream::operator<<(v10, &std::endl<char,std::char_traits<char>>);
      v11 = std::operator<<<std::char_traits<char>>(&std::cout, "1. View balance");
      std::ostream::operator<<(v11, &std::endl<char,std::char_traits<char>>);
      v12 = std::operator<<<std::char_traits<char>>(&std::cout, "2. View last operations");
      std::ostream::operator<<(v12, &std::endl<char,std::char_traits<char>>);
      v13 = std::operator<<<std::char_traits<char>>(&std::cout, "3. View info");
      std::ostream::operator<<(v13, &std::endl<char,std::char_traits<char>>);
```

По логике приложения после расшифровки мы получим полный доступ к кошельку, а следовательно и к флагу. Значит всё что нам надо это верно расшифровать кошелёк. При этом до функции расшифровки, всё что произошло - это запрос логина и пароля.

Функция расшифровки довольна большая, но самый главный момент заключён в начале (изображение ниже).

```
v31 = 0;
v25 = *std::vector<unsigned char,std::allocator<unsigned char>>::operator[](this + 160, 0LL);
v31 = 1;
if ( v25 != std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::size(this + 96) )
{
  v1 = 0;
}
else
{
  GenGamma(&v18, v25, v25);
  for ( i = 0; i < v25; ++i )
  {
    v2 = *std::vector<unsigned char,std::allocator<unsigned char>>::operator[](this + 160, v31 + i);
    v3 = *std::vector<int,std::allocator<int>>::operator[](&v18, i) ^ v2;
    if ( v3 != *std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::operator[](this + 96, i) )
    {
      v1 = 0;
      goto LABEL_26;
    }
  }
```

Вся проверка логина сводится к XOR’у с заранее сгенерированной гаммой, которая в качестве корня (seed) для генерации принимает размер логина. Данный размер сохраняется в первом байте файла. При этом пароль расшифровывается аналогичным образом.

Таким образом, мы можем очень просто вытащить пароль и логин из кошелька с помощью скрипта «solve.py», находящегося в директории solve.

```
➜ solve python solve.py test_wallet.acw
Username:  TPqMNSfS7RCOoU61o5
Password:  w71iEYoHAhYIjnnCALngv6hF9ZAC0ahxRGfjOhrroHxDBKxhLnbXYH
Use Username and Password for ./acw and view info!
➜ solve
```

Получаем логин и пароль. С помощью них расшифровываем кошелёк и читаем информацию.

```
➜ dev ./acw test_wallet.acw
Enter login: TPqMNSfS7RCOoU61o5
Enter password: w71iEYoHAhYIjnnCALngv6hF9ZAC0ahxRGfjOhrroHxDBKxhLnbXYH
-- Menu --
1. View balance
2. View last operations
3. View info
4. Exit
>> 3
Info: Cup{901caa40579a07ee5912514eebaf5526742ad03261971b233fd1cb88eee915ae}
-- Menu --
1. View balance
2. View last operations
3. View info
4. Exit
>>
```