Обеспечение безопасности работы сервисов КЦ

Существует несколько подходов к обеспечению безопасности работы сервисов кассового сервера. Ниже описаны три распространенных подхода.

Обеспечение безопасности конфигурационных файлов

Для обеспечения безопасности конфигурационных файлов необходимо запретить просмотр/редактирование конфигурационных файлов /opt/<название сервиса>/application.properties всем пользователям, кроме root, командой:

```
chmod 700 <путь к файлу/директории>
```

Операцию нужно повторить с конфигурационными файлами для всех установленных пакетов.

> Чтобы не запускать команду вручную, можно написать скрипт, который будет автоматически запускать команду для каждого конфигурационного файла.

Ограничение доступа к базам данных

MySQL

Ограничить доступ к БД можно несколькими способами:

- использовать в конфигурационном файле mysql настройку bind-address:
  - если требуется доступ извне, то необходимо записать в настройку значение 0.0.0.0,
  - если доступ к БД извне не требуется (контур локальный), то в настройку необходимо записать значение 127.0.0.1.
- использовать ролевую политику (**рекомендованный способ**):

Пример

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'netroot'@'%';
SHOW GRANTS for `netroot`@`%`;
+-----------------------------------+
| Grants for netroot@%              |
+-----------------------------------+
| GRANT USAGE ON *.* TO `netroot`@`%` |
+-----------------------------------+
1 row in set (0.00 sec)
```

Видно, что прав у пользователя с другого хоста нет (USAGE подразумевает отсутствие прав). При попытке подключиться с другого хоста будет выведена ошибка:

```
mysql> use artixcsAll;
ERROR 1044 (42000): Access denied for user 'netroot'@'%' to database 'artixcsAll'
```

Mongo

В целом процесс аналогичен описанному в подразделе MySQL. Чтобы ограничить доступ с других хостов, можно:

- указать в конфигурационном файле /etc/mongod.conf в настройке bindIp = 127.0.0.1 (localhost),
- использовать ролевую политику.

Пример

1. Создадим пользователя qwerty / qwerty с правами только на чтение.
2. В конфигурационном файле mongo укажем:

```
security:
  authorization: enabled
```

3. Перезапустим сервис.
4. Зайдем с указанием пользователя:

```
mongo -authenticationDatabase artixcs -u qwerty -p  --host 192.169.11.11 --port 27017
```

5. Добавим произвольную запись:

```
db.getCollection('serverInfo').insertOne({"versionRest": 192})
```

Получим ошибку:

```
WriteCommandError({
	"ok" : 0,
	"errmsg" : "not authorized on artixcs to execute command { insert: \"serverInfo\", ordered: true, $db: \"artixcs\" }",
	"code" : 13,
	"codeName" : "Unauthorized"
})
```

PostgreSQL

В целом процесс аналогичен описанному в подразделе MySQL:

- Для ограничения доступа к конфигурационным файлам необходимо использовать настройку listen_addresses. Подробнее об этом можно прочитать здесь.
- Для выдачи прав пользователям необходимо использовать соответствующие команды языка SQL. Подробнее об этом можно прочитать здесь и здесь.

Ограничение доступа через фаервол

Этот подход можно использовать только в том случае, когда описанные выше подходы не дали желаемого результата.

Для управления доступом к БД можно использовать настройки фаервола. Для этого необходимо:

1. Добавить правило, разрешающее доступ для localhost:

```
iptables -I INPUT -p tcp -s 127.0.0.1 --dport 3306 -j ACCEPT
```

2. Запретить всем доступ к порту 3306:

```
iptables -A INPUT -p tcp --dport 3306 -j DROP
```

3. При необходимости в начало цепочки добавить правило, разрешающее доступ избранным ip:

```
iptables -I INPUT -p tcp -s 192.169.11.11 --dport 3306 -j ACCEPT
```

В результате получится таблица с правилами для нужных ip:

```
1   ACCEPT     tcp  --  192.169.11.11       anywhere             tcp dpt:mysql
2   ACCEPT     tcp  --  192.169.11.111      anywhere             tcp dpt:mysql
3   DROP       tcp  --  anywhere            anywhere             tcp dpt:mysql
```

Безопасная авторизация

Для безопасной авторизации необходимо заменить логин и пароль по умолчанию во всех БД и сервисах с авторизацией по REST.

```
rest.port=38051
rest.host=localhost
rest.user=admin1
rest.password=admin1
```

Заменим в его конфигурационном файле логин (rest.user) и пароль (rest.password) до ядра КЦ. Если на ядре существует пользователь с таким логином и паролем, то сервис должен авторизоваться без ошибок.

Приведем список основных сервисов, имеющих настройки подключения к БД:

- artixcs-rest (mongodb, mysql, postgresql),
- artixcs-clickhouse-rest (mysql, postgresql),
- сервис обмена (nes),
- сервис tomcat8-artix,
- artixcs-datatransfer (mysql, mssql),
- artixcs-counters (postgresq),
- artixcs-undercut-asset,
- artixcs-online-card,
- accrual-bonus (доступ к БД счетчиков),
- artixcs-sales-ws,
- сервисы лояльности:
    - artixcs-accounting-coupons,
    - artixcs-accounting-bonuses,
    - artixcs-accounting-bonuses-certificates,
    - artixcs-accounting-certificates.

Чтобы изменить данные для подключения к БД (например, имя пользователя или хост), в /opt/artixcs-rest/application.properties необходимо добавить настройки (если по умолчанию они там отсутствуют):

Пример настройки для mysql

```
mysql.host=<хост>
mysql.port=<порт>
mysql.user=<логин>
mysql.password=<пароль>
```

В пароле для БД MySQL не рекомендуется использовать символы:

- {,
- },
- №.

Пример настройки для postgresql

```
postgresql.host=<хост>
postgresql.port=<порт>
postgresql.user=<логин>
postgresql.password=<пароль>
```

Такой подход работает для:

- сервисов artixcs-rest,
- сервисов лояльности:
  - artixcs-accounting-coupons,
  - artixcs-accounting-bonuses,
  - artixcs-accounting-bonuses-certificates,
  - artixcs-accounting-certificates.

В остальных сервисах настройки подключения уже указаны в /opt/<название сервиса>/application.properties со значениями по умолчанию, которые в целях безопасности рекомендуется изменить.

P.S. Также для более безопасной передачи данных на КЦ реализована возможность принимать продажи по протоколу https и отправлять https-запросы через сервис artixcs-rest-routing.