

Использование Рутокен в качестве защищенного хранилища

В некоторых случаях требуется организовать хранение небольших объемов секретных данных внутри энергонезависимой памяти Рутокен.

Наилучшим и наиболее стандартизованным способом доступа к хранилищу данных Рутокен является интерфейс PKCS#11 ([подробно](#)).

Объекты в Рутокен с точки зрения стандарта PKCS#11 выглядят следующим образом ([подробно](#)).

Уровни доступа к объектам в памяти Рутокен

Гостевой уровень позволяет просматривать, создавать, модифицировать и удалять только публичные (несекретные) объекты.

Для получения гостевого доступа не требуется знание ПИН-кода.

Пользовательский уровень позволяет просматривать, создавать, модифицировать и удалять как публичные (несекретные), так и приватные (секретные) объекты.

Пользовательский уровень предполагает знание ПИН-кода Пользователя.

Уровень Администратора (официера безопасности) предполагает знание ПИН-кода Администратора.

Администратор имеет доступ только к публичным объектам. Кроме того, Администратор обладает возможностью очистить память Рутокен, полностью включая приватные (секретные) объекты.

Типы объектов в хранилище данных Рутокен

Публичные (несекретные) объекты - объекты, видимые и доступные для чтения и записи с любым уровнем доступа.

Приватные (секретные) объекты - объекты, видимые и доступные для чтения и записи только с уровнем доступа "пользователь".

Двухфакторный доступ к секретной информации

Первый фактор - физическое наличие устройства.

Второй фактор - знание пользовательского пин-кода.

Пример записи файла через интерфейс PKCS#11

```

CK_BBOOL          ckTrue          =      CK_TRUE;
CK_BBOOL          ckFalse         =      CK_FALSE;
CK_OBJECT_CLASS   ocData          =      CKO_DATA;

CK_BYTE           ckLabel[]        =      {'f','i','l','e','n','a','m','e','.','t','x','t'};
CK_BYTE           ckValue[]        =      {'t','o','p','s','e','c','r','e','t','i','n','f','o'};

CK_UTF8CHAR       ckUserPIN[]      =      {'1','2','3','4','5','6','7','8'};

CK_ATTRIBUTE       DataObject[]     =      {
    {CKA_CLASS,      &ocData,      sizeof(CK_OBJECT_CLASS) },    //
    {CKA_TOKEN,      &ckTrue,      sizeof(CK_BBOOL) },           //
    {CKA_PRIVATE,    &ckTrue,      sizeof(CK_BBOOL) },           // PIN-
    {CKA_LABEL,      ckLabel,      sizeof(ckLabel) },            //
    {CKA_VALUE,      ckValue,      sizeof(ckValue) },            //
};

CK_RV              rv;

CK_SESSION_HANDLE  hSession;
CK_OBJECT_HANDLE   ckHandle;

.
.
rv = functionList->C_Login(hSession,
                           CKU_USER,
                           ckUserPIN,
                           sizeof(ckUserPIN));

.
.
rv = functionList->C_CreateObject(hSession,
                                  &DataObject,
                                  sizeof(DataObject)/sizeof(CK_ATTRIBUTE),
                                  &ckHandle);

.
.

```

Пример чтения файла через интерфейс PKCS#11

```

/*****
 *
 *****/
static int findObjects(CK_FUNCTION_LIST_PTR functionList, // PKCS#11
                      CK_SESSION_HANDLE session,        //
                      CK_ATTRIBUTE_PTR attributes,       //
                      CK_ULONG attrCount,               //
                      CK_OBJECT_HANDLE_PTR* objects,     //
                      CK_ULONG* objectsCount            //
)
{
    CK_RV rv; // . , PKCS#11
    CK_ULONG newObjectsCount; // , C_FindObjects
    CK_ULONG bufferSize; //
    CK_OBJECT_HANDLE_PTR buffer; // , realloc
    int errorCode = 1; //

    /*****
     *
     *****/
    rv = functionList->C_FindObjectsInit(session, attributes, attrCount);
    if (rv != CR_OK) goto exit;

    /*****
     *
     *****/
    *objects = NULL;
    *objectsCount = 0;
}

```

```

for (bufferSize = 8;; bufferSize *= 2) {
    buffer = (CK_OBJECT_HANDLE_PTR)realloc(*objects, bufferSize * sizeof(CK_OBJECT_HANDLE));
    if (buffer == NULL) goto find_final;
    *objects = buffer;

    rv = functionList->C_FindObjects(session, *objects + *objectsCount, bufferSize - *objectsCount,
&newObjectsCount);
    if (rv != CR_OK) goto find_final;

    *objectsCount += newObjectsCount;

    if (*objectsCount < bufferSize) {
        break;
    }
}

/*****
*
*
*****/
if (*objectsCount != 0) {
    buffer = (CK_OBJECT_HANDLE_PTR)realloc(*objects, *objectsCount * sizeof(CK_OBJECT_HANDLE));
    if (buffer == NULL) goto find_final;
    *objects = buffer;
}

errorCode = 0;

/*****
*
*
*****/
find_final:
rv = functionList->C_FindObjectsFinal(session);
if (rv != CR_OK) errorCode = 1;

/*****
*
*
*****/
if (errorCode || *objectsCount == 0) {
    free(*objects);
    *objects = NULL_PTR;
}

exit:
return errorCode;
}

```

```

CK_OBJECT_CLASS ocData = CKO_DATA;
CK_BYTE label[] = {'f','i','l','e','n','a','m','e','.','t','x','t'};;
CK_BBOOL ckTrue = CK_TRUE;
CK_BBOOL ckFalse = CK_FALSE;

CK_OBJECT_HANDLE_PTR objects; //
CK_ULONG objectCount; //

CK_RV rv = CKR_OK;

CK_UTF8CHAR ckUserPIN[] = {'1','2','3','4','5','6','7','8'};

CK_ATTRIBUTE attrDataReadTpl[] = {
    {CKA_CLASS, &ocData, sizeof(ocData)},
    {CKA_LABEL, label, sizeof(label) - 1},
    {CKA_PRIVATE, &ckTrue, sizeof(ckTrue)}, // PIN-
    {CKA_TOKEN, &ckTrue, sizeof(ckTrue)} //
};

rv = C_Login(hSession,
    CKU_USER,
    ckUserPIN,
    sizeof(ckUserPIN));
if (rv != CKR_OK)
    return;

rv = findObjects(functionList, session, attrDataReadTpl, arraysize(attrDataReadTpl), &objects, &objectCount);
.
.

if (objectCount == 1) {
    CK_ATTRIBUTE attrValue = { CKA_VALUE, NULL_PTR, 0 };

    rv = functionList->C_GetAttributeValue(session, //
        objects[0], //
        &attrValue, //
        1); //

    if (rv != CKR_OK)
        return;

    /* */
    attrValue.pValue = (CK_BYTE*)malloc(attrValue.ulValueLen);
    if (attrValue.pValue == NULL)
        return;

    memset(attrValue.pValue, 0, (attrValue.ulValueLen * sizeof(CK_BYTE)));

    /* */
    printf("Getting object value");
    rv = functionList->C_GetAttributeValue(session, //
        objects[0], //
        &attrValue, //
        1); //

    if (rv != CKR_OK)
        return;

    /* */
    printf("Data is:\n");

    printf("%.s", attrValue.ulValueLen, (const char*)attrValue.pValue);

    free(attrValue.pValue);
}

```

