TeleLock API

Dokumentacja programisty
API biblioteki tlapi_debug.dll

TeleVox SOFTWARE & SECURITY

Wstęp

Biblioteka tlapi_debug.dll jest częścią oprogramowania TeleLock. Oprogramowanie to umożliwia zabezpieczanie plików wykonywalnych (programów) przed nielicencjonowanym użyciem, z wykorzystaniem TeleTokenu. Sam program TeleLock zabezpiecza programy poprzez ich „owijanie" (inaczej kopertowanie) i szyfrowanie części wykonywalnego kodu. Osoba zabezpieczająca decyduje o wprowadzeniu ew. licencji czasowej, lub na ilość uruchomień programu. Natomiast biblioteka tlapi_debug.dll wprowadza dodatkowe możliwości do wykorzystania przez programistę zabezpieczającego swój program, poprzez dostarczenie mu prostego w użyciu API, pozwalającego na korzystanie z dodatkowych możliwości oferowanych przez TeleToken. Są to m.in. odczyt i zapis pamięci TeleTokenu (wew. szyfrowanej), korzystanie ze sprzętowego szyfrowania i deszyfrowania AES, jak również z liczników zawartych w TeleTokenie. Taka funkcjonalność pozwala na indywidualne zaprojektowanie zabezpieczenia dostosowanego do konkretnych potrzeb, a proste API pozwala na skupienie się na innych istotnych zagadnieniach.

Możliwości i ograniczenia TeleLock API

TeleLock API dostarcza programiście łatwego sposobu komunikacji z TeleTokenem w postaci biblioteki tlapi_debug.dll. Jest ono pomyślane w ten sposób, aby wyeliminować konieczność korzystania z pełnego API TeleTokenu i zarazem dać możliwość korzystania z samego TeleTokenu jednocześnie nie rezygnując z zabezpieczenia oferowanego przez TeleLock (jak było to w poprzedniej jego wersji).
Uwaga – korzystanie z pełnego API TeleTokenu wraz z zabezpieczeniem TeleLock jest w dalszym ciągu nie zalecane i nie wspierane.
Pośród funkcji oferowanych przez TeleToken i udostępnionych w TeleLock API znajdują się:

1. Zapis i odczyt pamięci TeleTokenu.
   Pamięć TeleTokenu zorganizowana jest w sektory o długości 16 bajtów każdy. Sektory mogą być zapisywane/odczytywane tylko i wyłącznie w całości. W celu ułatwienia dostępu programiście w tym API zostały wprowadzone dodatkowo rozkazy odczytu i zapisu danych typu bajt czy boolean (pojedynczy bit), poprzez zmapowanie ich na pierwsze sektory pamięci. Rozkazy te oferują adresowanie z zakresu 0 – 255, czyli za pomocą operacji o wielkości bajt mamy dostęp do pierwszych 16 sektorów pamięci (16 * 16 = 256), a za pomocą operacji o wielkości bit do pierwszych 2 sektorów (2 * 16 * 8 = 256). Oczywiście dane zapisane w ten sposób wpłyną na dane odczytywane jako cały sektor.
   Dostępna wielkość pamięci w zależności od modelu TeleTokenu kształtuje się jak poniżej:

Typ TeleTokenu	Pamięć w sektorach	Pamięć w bajtach
TeleToken AES	16	256
TeleToken AES Pro	72	1152

   TeleToken oferuje możliwość protekcji każdego z dostępnych sektorów pamięci osobno. Powoduje to ustawienie ich w tryb tylko do odczytu i zablokowanie możliwości zapisu. Jedyną metodą pozwalającą na ponowny zapis zaprotegowanego sektora jest sformatowanie TeleTokenu, co powoduje utratę wszystkich danych w nim zawartych i przywrócenie

konfiguracji fabrycznej.
TeleToken przygotowany za pomocą programu TeleLock oferuje programiście korzystającemu z TeleLock API całą dostępną pamięć. Po zaprogramowaniu przez TeleLock pamięć jest pusta (wypełniona zerami), a wszystkie sektory mają wyzerowane bity protekcji. Numerowanie sektorów rozpoczyna się od indeksu „0".

2. Liczniki sprzętowe.
   W zależności od modelu TeleToken zawiera zestaw liczników sprzętowych:

Typ TeleTokenu	Ilość liczników	Ilość liczników dostępnych poprzez API
TeleToken AES	8	4
TeleToken AES Pro	32	28

   Ze względu na wykorzystanie dla wewnętrznych celów TeleLock, dwa z liczników TeleTokenu są niedostępne poprzez TeleLock API. Numerowanie liczników rozpoczyna się od indeksu „0".

   Liczniki oferują programiście możliwość nadzoru czasowego/ilościowego. Każdy licznik może przechowywać wartość 2 – bajtową, czyli możliwe są wartości z zakresu 0 – 65535. Odczyt liczników może być wykonywany w jednym z trybów:

   - Odczyt bez zmiany zawartości licznika
   - Odczyt ze zmniejszeniem o 1 wartości licznika
   - Odczyt ze zwiększeniem o 1 wartości licznika

   W przypadku używania rozkazów zmniejszających/zwiększających zawartość licznika należy zwrócić uwagę na fakt, że licznik „zatrzymuje" się pod osiągnięciu wartości „0". Dalsze odczyty ze zwiększaniem/zmniejszaniem wartości zawsze będą zwracały „0", aż do ponownego ustawienia licznika wartością niezerową.

3. Szyfrowanie i deszyfrowanie.
   TeleToken dzięki wbudowanym algorytmom AES oferuje sprzętową możliwość szyfrowania i deszyfrowania danych. Ponieważ zaimplementowany wewnątrz TeleTokenu algorytm operuje na danych 128-bitowych (czyli 16-bajtowych) operacje szyfrowania i deszyfrowania wykonywane są bezpośrednio na danych o takiej wielkości. W celu ułatwienia tworzenia oprogramowania w TeleLock API zostały wprowadzone szyfry blokowe, operujące na danych dowolnej długości. Działanie tych funkcji polega na wstępnym przygotowaniu klucza wewnątrz TeleTokenu (dla większego bezpieczeństwa), a następnie wykonania operacji szyfrowania/deszyfrowania blokowego kluczem symetrycznym w bibliotece (dla szybkości operacji).

   Hasła szyfrowania (również 128-bitowe) są składowane w TeleTokenie w sposób bezpieczny, uniemożliwiający ich odczytanie. Ilość dostępnych haseł dla poszczególnych modeli TeleTokenów kształtuje się następująco:

Typ TeleTokenu	Ilość haseł	Ilość haseł dostępnych poprzez API
TeleToken AES	8	4
TeleToken AES Pro	32	28

Tryby działania biblioteki

Biblioteka tlapi_debug.dll posiada dwa, odrębne tryby działania.

Pierwszym jest tryb „debug”, stworzony do wykorzystywania przez programistę podczas rozwijania swojego oprogramowania. Biblioteka w tym trybie to dostarczony z pakietem TeleLock plik „tlapi_debug.dll”. Zalecane jest jego umieszczenie w katalogu tworzonego programu, tak, aby podczas testowania programu miał on do niej dostęp. Biblioteka w tym trybie spełnia w pewnym sensie rolę „emulatora”, gdyż nie pozwala na korzystanie z prawdziwego TeleTokenu (w postaci sprzętowego zabezpieczenia podłączanego do portu USB). W zamian wszystkie funkcje oferowane przez bibliotekę wykonywane są programowo, w jej obrębie. W trybie tym funkcje przechowywania wewnętrznej zawartości pamięci TeleTokenu przejmują zwykłe pliki binarne, umiejscowione w katalogu biblioteki (Patrz: *Użycie biblioteki*). Uwaga – zawartość tych plików nie jest szyfrowana, ani w żaden inny sposób zabezpieczona.

Drugim trybem jest właściwy tryb zabezpieczenia. Podczas zabezpieczania gotowego programu przy pomocy TeleLock wszystkie odwołania do biblioteki tlapi_debug.dll zostają przechwycone i zamienione na wewnętrzne funkcje zabezpieczenia. Cała funkcjonalność i API tej biblioteki zostają umieszczone wewnątrz zabezpieczanej aplikacji. Od tej pory ma ona możliwość komunikacji z właściwym TeleTokenem i wykonywania operacji przy jego wykorzystaniu. Biblioteka w postaci pliku „tlapi_debug.dll” nie powinna być oczywiście rozpowszechniania razem z zabezpieczonym programem, jest to niepotrzebne. Wymagane jest tylko dołączenie biblioteki „TeleToken.dll”.

Takie rozdzielenie biblioteki zostało podyktowane uzyskaniem dobrego poziomu bezpieczeństwa przy jednoczesnym umożliwieniu programiście tworzącemu swoje oprogramowanie korzystania z trybu „debug”.

Różnice między działaniem w obu trybach

Niektóre funkcje działają w trybie „debug” w sposób ograniczony. Do takich funkcji należy protekcja sektorów pamięci. W trybie „debug” wywołanie rozkazu protekcji zwraca co prawda TRUE, ale nie blokuje możliwości zapisu.

Użycie biblioteki

Do pakietu zostały dołączone pliki nagłówkowe dla C/C++ oraz Delphi. Dla C/C++ dodatkowo także plik „tlapi_debug.lib” do dołączenia do programu.

Biblioteka (plik „tlapi_debug.dll”) powinna znaleźć się w katalogu uruchamiania rozwijanej/testowanej wersji tworzonego programu. W tym samym katalogu mogą pojawić się dodatkowe pliki, przechowujące zawartość TeleTokenu, w zastępstwie sprzętu. Pliki te mogą zostać wyeksportowane przez program TeleLock (Patrz: *TeleLock – dokumentacja* programu). W ten sposób można korzystać dokładnie z tych samych danych TeleTokenu w trybie „debug”, jak i później w produkcie finalnym. Pliki te to:

1. tlapi_debug.mem – emulujący magazyn wewnętrznej pamięci TeleTokenu
2. tlapi_debug.psw – zawierający hasła szyfrowania do użycia w funkcjach tego API (Uwaga – hasła nie są szyfrowane, ani w żaden inny sposób zabezpieczone)
3. tlapi_debug.cnt – przechowujący zawartość emulowanych liczników TeleTokenu

Opis funkcji API

Funkcja TLA_Open

Składnia

```
BOOL TLA_Open(void);
```

Argumenty

Brak

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Uwagi

Wywołanie tej funkcji jest niezbędne w celu nawiązania komunikacji z TeleTokenem. Należy ją więc wywołać przed próbą wykonania jakiejkolwiek operacji za pomocą TeleTokenu. Również, jeśli któraś z pozostałych funkcji zwróciła FALSE (czyli jej wywołanie nie powiodło się), należy ponownie wywołać TLA_Open, aby upewnić się, że komunikacja z TeleTokenem nie została przerwana.

Funkcja TLA_Close

Składnia

```
void TLA_Close(void);
```

Argumenty

Brak

Wartość zwracana

Brak

Uwagi

Ta funkcja zamyka komunikację z TeleTokenem. Można jej użyć w dowolnym momencie w aplikacji, jeśli nie istnieje już potrzeba odwołań do TeleTokenu. Można również wznowić komunikację za pomocą TLA_Open.

Funkcja TLA_TokenVersion

Składnia

```
enum TLA_TOKENVER TLA_TokenVersion(void);
```

Argumenty

Brak

Wartość zwracana

Zwracana wartość ma typ enumeracyjny TLA_TOKENVER i może przyjmować następujące wartości:

Zdefiniowana wartość	Znaczenie
TLA_TV_UNKNOWN	Nierozpoznany typ TeleTokenu
TLA_TV_AES	TeleToken AES
TLA_TV_AES_PRO	TeleToken AES Pro

Uwagi

Funkcja ta pozwala na sprawdzenie jaki konkretny typ ma podłączony TeleToken. W rezultacie można w ten sposób określić, ile np. komórek pamięci, czy haseł jest dostępnych w TeleTokenie. Szczegóły dotyczące funkcji oferowanych przez konkretne wersje TeleTokenu znajdują się w jego dokumentacji.

Funkcja TLA_ReadSector

Składnia

```
BOOL TLA_ReadSector(BYTE index, BYTE data[16]);
```

Argumenty

index [in]
: Indeks sektora pamięci do odczytu.

data [out]
: Bufor do którego zostanie zapisany odczytany sektor pamięci.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Uwagi

Sektory numerowane są od „0”. Proszę kierować się dokumentacją TeleTokenu odnośnie dokładnej ilości sektorów pamięci w zakupionym modelu. Próba odczytu sektora spoza zakresu zwróci FALSE.

Dane są dodatkowo, wewnętrznie szyfrowane, zanim zostaną zapisane w TeleTokenie. Uniemożliwia to odczyt jego zawartości za pomocą TeleToken API, nawet w przypadku zdobycia haseł komunikacyjnych.

Funkcja TLA_WriteSector

Składnia

```
BOOL TLA_WriteSector(BYTE index, BYTE data[16]);
```

Argumenty

index [in]
: Indeks sektora pamięci do zapisu.

data [in]
: Bufor którego zawartość zostanie zapisana do wybranego sektora pamięci.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Funkcja TLA_ProtectSector

Składnia

```
BOOL TLA_ProtectSector(BYTE index);
```

Argumenty

index [in]
: Indeks sektora pamięci do zabezpieczenia przed zapisem.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Uwagi

Proszę pamiętać, że zaprotegowany sektor może zostać ponownie zapisany tylko i wyłącznie po wykonaniu formatowania TeleTokenu. Można go więc traktować jako sektor tylko do odczytu.

Jeśli sektor jest mapowany do odczytu bajtowego lub typu boolean, także te operacje (TLA_WriteByte, TLA_WriteBoolean) zostaną zablokowane.

Funkcja TLA_ReadCounter

Składnia

```
BOOL TLA_ReadCounter(BYTE index, enum TLA_CNTCMDS cmd, USHORT *value);
```

Argumenty

index [in]

Indeks licznika do odczytu.

cmd [in]

Tryb odczytu. Możliwe są następujące wartości:

Zdefiniowana wartość	Znaczenie
TLA_CNT_READ	Odczyt licznika
TLA_CNT_INC_READ	Odczyt licznika ze zwiększeniem o 1
TLA_CNT_DEC_READ	Odczyt licznika ze zmniejszeniem o 1

value [out]

Wskaźnik na zmienną do której zostanie zapisana odczytana wartość licznika.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Uwagi

Liczniki numerowane są od „0”.

Funkcja TLA_WriteCounter

Składnia

```
BOOL TLA_WriteCounter(BYTE index, USHORT value);
```

Argumenty

index [in]

Indeks licznika do zapisu.

value [in]

Wartość do zapisu do wybranego licznika.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Funkcja TLA_ReadBoolean

Składnia

```
BOOL TLA_ReadBoolean(BYTE index, BOOL *value);
```

Argumenty

index [in]
: Indeks bitu do odczytu.

value [out]
: Wskaźnik na zmienną do której zostanie zapisana odczytana wartość bitu.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Funkcja TLA_WriteBoolean

Składnia

```
BOOL TLA_WriteBoolean(BYTE index, BOOL value);
```

Argumenty

index [in]
: Indeks bitu do zapisu.

value [in]
: Wartość do zapisu do wybranego bitu.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Funkcja TLA_ReadByte

Składnia

```
BOOL TLA_ReadByte(USHORT index, BYTE *value);
```

Argumenty

index [in]
: Indeks bajta do odczytu.

value [out]
Wskaźnik na zmienną do której zostanie zapisana odczytana wartość bajta.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Funkcja TLA_WriteByte

Składnia

```
BOOL TLA_WriteByte(USHORT index, BYTE value);
```

Argumenty

index [in]
Indeks bajta do zapisu.

value [in]
Wartość do zapisu do wybranego bajta.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Funkcja TLA_Encrypt

Składnia

```
BOOL TLA_Encrypt(BYTE index, BYTE data[16]);
```

Argumenty

index [in]
Indeks hasła do użycia przy szyfrowaniu.

value [in/out]
Wskaźnik na bufor na którym zostanie przeprowadzony rozkaz szyfrowania wybranym hasłem TeleTokenu.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Uwagi

Hasła numerowane są od „0”. Ilość dostępnych haseł zależy od konkretnego typu TeleTokenu AES. Szyfrowanie przeprowadzane jest „w miejscu”, czyli bufor jest zarówno argumentem wejściowym, jak i wyjściowym.

Funkcja TLA_Decrypt

Składnia

```
BOOL TLA_Decrypt(BYTE index, BYTE data[16]);
```

Argumenty

index [in]
: Indeks hasła do użycia przy deszyfrowaniu.

value [in/out]
: Wskaźnik na bufor na którym zostanie przeprowadzony rozkaz deszyfrowania wybranym hasłem TeleTokenu.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Uwagi

Patrz „TLA_Encrypt”.

Funkcja TLA_EncryptBlock

Składnia

```
BOOL TLA_EncryptBlock(BYTE index, BYTE *dataIn, DWORD length, BYTE *dataOut);
```

Argumenty

index [in]
: Indeks hasła do użycia przy szyfrowaniu.

dataIn [in]
: Bufor z danymi do zaszyfrowania.

length [in]
: Długość danych do zaszyfrowania.

dataOut [out]
: Wskaźnik na bufor do którego zostaną zapisane zaszyfrowane dane. Bufor musi posiadać miejsce na przyjęcie danych o długości: dane wejściowe + 32 bajty.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Uwagi

Funkcja służy do szyfrowania danych o dowolnej długości przy użyciu szyfru blokowego. Wielkość zaszyfrowanych danych zawsze jest dłuższy o dodatkowe 32 bajty względem danych wejściowych. Taki też blok danych należy w całości przekazać do funkcji „TLA_DecryptBlock" w celu jego odszyfrowania.

Funkcja TLA_DecryptBlock

Składnia

```
BOOL TLA_DecryptBlock(BYTE index, BYTE *dataIn, DWORD length, BYTE *dataOut);
```

Argumenty

index [in]
: Indeks hasła do użycia przy deszyfrowaniu.

dataIn [in]
: Bufor z danymi do odszyfrowania.

length [in]
: Długość danych do odszyfrowania. Jest to długość całego zaszyfrowanego uprzednio bloku (wliczając dodatkowe 32 bajty).

dataOut [out]
: Wskaźnik na bufor do którego zostaną zapisane odszyfrowane dane. Bufor musi posiadać miejsce na przyjęcie danych o długości: dane wejściowe (zaszyfrowane) - 32 bajty.

Wartość zwracana

TRUE jeśli funkcja wykona się poprawnie, FALSE w przeciwnym wypadku.

Uwagi

Funkcja służy do odszyfrowania danych, które zostały zaszyfrowane uprzednio za pomocą funkcji „TLA_EncryptBlock". Długość odszyfrowanych danych będzie o 32 bajty mniejsza od danych wejściowych, czyli równa będzie ilości danych przed zaszyfrowaniem.

ul. Dobrego Pasterza 100
31-416 Kraków

tel (012) 415-67-00
fax (012) 415-67-15
e-mail: biuro@televox.pl
http://www.televox.pl