Форматы данных PKI

Содержание

Введение

Этот документ описывает наиболее распространенные форматы данных Инфраструктуры открытых ключей (PKI) и кодировки.

Предварительные условия

Требования

Компания Cisco рекомендует предварительно ознакомиться со следующими предметами:

- криптография общего ключа (базовые понятия).
- инфраструктура открытого ключа (базовые понятия).

Используемые компоненты

Настоящий документ не имеет жесткой привязки к каким-либо конкретным версиям программного обеспечения и оборудования.

Сведения, представленные в этом документе, были получены от устройств, работающих в специальной лабораторной среде. Все устройства, описанные в этом документе, были запущены с чистой (стандартной) конфигурацией. В рабочей сети необходимо изучить

потенциальное воздействие всех команд до их использования.

Условные обозначения

Сведения об условных обозначениях см. в документе Условные обозначения технических терминов Cisco.

Нотация ASN.1

Abstract Syntax Notation One (ASN.1) является формальным языком для определения типов данных и значений, и как те типы данных и значения используются и объединяются в различных структурах данных. Цель стандарта состоит в том, чтобы определить абстрактный синтаксис информации, не ограничивая, как информация закодирована для передачи.

Вот пример, выбранный от *RFC X.509*:

```
   Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }
CertificateSerialNumber ::= INTEGER
Validity ::= SEQUENCE {
notBefore Time,
notAfter Time }
Time ::= CHOICE {
utcTime UTCTime,
generalTime GeneralizedTime }
```

См. эти документы от Международного союза электросвязи (ITU-T) узлы стандартов:

- X . 680 ASN.1: Спецификация основной нотации
- X . 681 ASN.1: спецификация Информационного объекта
- X . 682 ASN.1: Ограничительная спецификация
- X . 683 ASN.1: Параметризация спецификаций ASN.1

Поиск рекомендаций ITU-T - Поиск **X.509** в **Rec. или стандарте** с **Языковым** набором к **ASN.1**.

Кодировка BER/CER/DER

ITU-T определил стандартный способ закодировать структуры данных, описанные в ASN.1 в двоичные данные. X . 690 определяет Базовые правила кодирования (BER) и его два подмножества, Канонические правила кодирования (CER) и Выдающиеся правила кодирования (DER). Все три основываются на полях данных **Type Length Value**, упакованных в иерархической структуре, которая создана из **ПОСЛЕДОВАТЕЛЬНОСТЕЙ**, **НАБОРОВ** и **ВЫБОРОВ**, с этими различиями:

- BER предоставляет несколька способов закодировать те же данные, которые не подходят для крипто-операций.
- CER предоставляет определенное кодирование и использует неопределенные данные длины с маркером конца данных в конкретных случаях.
- DER предоставляет определенное кодирование и использует явные метки длины в

конкретных случаях.
- Среди этих трех DER является тем, с которым обычно встречаются при контакте с PKI и крипто-информационными наполнениями.

Пример: В DER 20-разрядное значение `1010 1011 1101 1100 года 1110` закодирован как :

- **метка**: `0x03` (строка битов)
- **длина:** `0x04` (байты)
- **значение**: `0x04ABCDE0`
- **завершенное кодирование DER**: `0x030404ABCDE0`

Продвижение **`04`** означает, что последние 4 бита (равняется запаздыванию **`0`** цифра) закодированного значения должен быть сброшен, потому что закодированное значение не заканчивается на границе в байтах.

См. эти документы от узла стандартов TUT:

- Правила кодирования X . 690 ASN.1: Спецификация Базовых правил кодирования (BER), Канонических правил кодирования (CER) и Выдающихся правил кодирования (DER)

От сайта Википедии обратитесь к этим документам:

- Базовые правила кодирования
- Канонические правила кодирования
- Выдающиеся правила кодирования

Шестнадцатеричный дамп DER

Cisco IOS, Устройство адаптивной защиты (ASA) и другие устройства отображают содержание DER как **шестнадцатеричный дамп** с **командой show running-config**. Вот выходные данные:

```
crypto pki certificate chain root
certificate ca 01
30820213 3082017C A0030201 02020101 300D0609 2A864886 F70D0101 04050030
1D310C30 0A060355 040B1303 54414331 0D300B06 03550403 1304726F 6F74301E
170D3039 30373235 31313436 33325A17 0D313230 37323431 31343633 325A301D
...
```

Этот вид шестнадцатеричного дампа может быть преобразован назад в DER в различных способах. Например, можно удалить пробелы и передать его по каналу к **xxd программе**:

```
$ cat ca.hex | tr -d ' ' | xxd -r -p -c 32 | openssl x509 -inform der -text -noout
```

Другой простой способ должен использовать этот сценарий Perl :

```
#!/usr/bin/perl
foreach (<>) {
s/[^a-fA-F0-9]//g;
print join("", pack("H*", $_));
} $ perl hex2der.pl < hex-file.txt > der-file.der
```

Кроме того, компактный способ преобразовать **дампы свидетельства**, каждый ранее вручную скопированный к файлу с расширением **.hex**, из командной строки **удара** как показано здесь:

```
for hex in *.hex; do
b="${hex%.hex}"
hex2der.pl < "$hex" > "$b".der
openssl x509 -inform der -in "$b".der > "$b".pem
openssl x509 -in "$b".pem -text -noout > "$b".txt
done
```

Каждый файл приводит к:

- **file.hex -** Исходный файл (должен содержать только шестнадцатеричные числа).
- **file.der -** Сертификат в DER (двоичный) формат.
- **file.pem -** Сертификат в PEM (Base64 + заголовок/нижний колонтитул) формат.
- **file.txt -** Удобная для пользователя, читаемая версия сертификата.

Кодирование base64

Кодирование Base64 представляет двоичные данные только с 64 печатаемыми символами (`A-Za-z0-9+/`) так же **кодировать программой uuencode**. В преобразовании от двоичных файлов до Base64 каждый 6-разрядный блок исходных данных закодирован в 8-разрядный печатаемый ASCII - символ с таблицей преобразования. Поэтому размер данных после кодирования увеличился на 33 процента (времена данных 8 разделенных на 6 битов, равняется 1.333).

24-разрядный буфер используется для трансляции трех (3) групп из восьми (8) биты в четыре (4) группы шести (6) битов. Поэтому один (1) или два (2) байта заполнения данных могли бы требоваться в конце входящего потока данных. Заполнение обозначено в конце закодированных Base64 данных, каждый равняется (`=`) распишитесь за каждую группу из восьми (8) биты заполнения, добавленные к вводу во время кодирования.

См. данный пример из Википедии.

См. новую информацию в RFC 4648: Base16, Base32 и Кодировка Данных Base64.

Кодирование PEM

Privacy Enhanced Mail (PEM) является Engineering Task Force полнофункционального Интернета (IETF) стандарт PKI для обмениваний безопасными сообщениями. Это широко больше не используется как таковое, но его синтаксис инкапсуляции был широко одолжен, чтобы отформатировать и обмениваться закодированными Base64 связанными с PKI данными.

RFC PEM 1421, разделите 4.4: Механизм инкапсуляции, определяет сообщения PEM, как разграничено Границами Инкапсуляции (EBS), которые основываются на RFC 934 с этим форматом:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Header: value
Header: value
...

Base64-encoded data
...
-----END PRIVACY-ENHANCED MESSAGE-----
```

На практике сегодня, когда форматированные данные PEM распределены, этот граничный формат используется:

```
-----BEGIN type-----
...
-----END type-----
```

тип может быть с другими ключами или сертификатами, такими как:

- `RSA PRIVATE KEY`
- `ENCRYPTED PRIVATE KEY`
- `CERTIFICATE`
- `CERTIFICATE REQUEST`
- `X509 CRL`

Примечание: Несмотря на то, что RFC не делают это обязательным, количество продвижения и запаздывающих тире (-) в EBS является значительным и должен всегда быть пять (5). В противном случае некоторые приложения, такие как OpenSSL, дросселируют на вводе. С другой стороны, другие приложения, такие как Cisco IOS, не требуют EBS вообще.

См. эти новые RFC для получения дополнительной информации:

- RFC 1421 : первая часть PEM: передайте шифрование и процедуры проверки подлинности
- RFC 1422 : вторая часть PEM: основанное на сертификате управление ключами
- RFC 1423 : часть III PEM: алгоритмы, режимы и идентификаторы
- RFC 1424 : IV части PEM: ключевая сертификация и связанные сервисы

X . 509 сертификатов и CRL

X . 509 является подмножеством X.500, который является расширенной спецификацией ITU о Взаимодействии открытых систем. Это имеет дело в частности с сертификатами и открытыми ключами и было адаптировано как интернет-стандарт IETF. X . 509 предоставляет структуру и синтаксис, выраженный в RFC Нотацией ASN.1, чтобы хранить информацию сертификата и списки отозванных сертификатов.

В PKI X.509 CA выполняет сертификат, который связывает открытый ключ, например: Ривест-Шамир-Адлемен (RSA) или ключ Алгоритма цифровой подписи (DSA) к определенному Составному имени (DN), или к альтернативному названию, такому как адрес электронной почты или полное доменное имя (FQDN). DN придерживается структуры в стандартах X.500. Например:

```
CN=common-name, OU=organizational-unit, O=organization, L=location,
C=country
```

Из-за определения ASN.1 данные X.509 могут быть закодированы в DER, чтобы обменяться в двоичной форме, и дополнительно, преобразовать в Base64/PEM для основанных на тексте средств сообщения, таких как вставка копии на терминале.

- См. этот документ X.509 стандартов ITU-T Взаимодействие открытых систем - Каталог: Общий ключ и платформы сертификата атрибута.
- См. RFC 5280: X . 509 Профилей Сертификата и Списка отозванных сертификатов (CRL) для получения дополнительной информации.

Стандарты PKCS

Стандарты криптографии общего ключа (PKCS) являются спецификациями от лабораторных работ RSA, которые частично развились в промышленные стандарты. Те, с которыми чаще всего встречаются, имейте дело с этими темами; однако, не все они имеют дело с форматами данных.

PKCS#1 (RFC 3347) - Покрывает аспекты реализации основанной на RSA криптографии (крипто-примитивы, схемы шифрования/подписи, синтаксис ASN.1).

PKCS#5 (RFC 2898) - Покрывает основанную на пароле ключевую деривацию.

PKCS#7 (RFC 2315) и S/MIME RFC 3852 **-** Определяет синтаксис сообщения для передачи подписанный и зашифрованные данные и отнесенные сертификаты. Часто используемый просто в качестве контейнера для сертификатов X.509.

PKCS#8-Определяет синтаксис сообщения для переноса открытого текста или зашифрованных криптографических пар RSA.

PKCS#9 (RFC 2985) - Определяет дополнительные классы объектов и идентификационные атрибуты.

PKCS#10 (RFC 2986) - Определяет синтаксис сообщения для Запросов подписи сертификата (CSR). CSR передается объектом CA и содержит информацию, которая будет подписана CA, таким как информация с открытым ключом, идентичность и дополнительные атрибуты.

PKCS#12 - Определяет контейнер для упаковки связанных данных PKI (как правило, **пара ключей объекта + свидетельство объекта + корневые сертификаты CA и промежуточные сертификаты CA**) в отдельном файле. Это - развитие Exchange Личных данных Microsoft (PFX) формат.

См. эти ресурсы:

- Статья Wikipedia относительно PKCS
- Страница RSA Labs на PKCS

Дополнительные сведения

- Cisco Systems – техническая поддержка и документация