9)

**формат OSCP-відповіді**  
  
Відповідь складається 2 полів:

OSCP Response = SEQ<respStatus Status, respBytes [0] RespBytes OPT>

Друге наявне тоді і лише тоді коли статус ок або дорівнює 0.

Типи статусу Status = ENUMERATED :

* 0 – ok – ок, все добре
* 1 – bad request – некоректни формат запиту
* 2 – internal server error – внутрішня помишла серверу OA
* 3 – try later
* 4 – ця відповідь не використовується
* 5 – запит отрібно підписати
* 6 – клієнт або респондер не мають повноважень

Відповідь отримуємо лише якщо статус дорінює 0, а сама відповідб теж складається з двох полів Response Bytes = SEQ {

* respType OID – тип відповіді
* response OCTET STRING – кодована відповідь }  
  Наразі існує 1 стандартний формат OCSP відповідей який називається Basic OCSP Response і має ідентифікатор 1.3.6.1.5.5.7.48.1.1

Стандартна підписана структура Basic OCSP Response = SEQ{

* tbsResponceData ResponceData – дані над якими будемо ставити підпис
* signal AlgId – алгоритм підпису
* sign BIT STRING- сам підпис
* certs [0]SEQ of Certificates OPT – набір сертифікатів  
  }

ResponceData = SEQ{

* version [0]INTEGER (DEFAULT = v1) – версія
* responcerID RespID – ідентифікатор відповідача/респондера
* producedAt GenerilizedTime – час коли була сформована відповідь
* responses SEQ OF SingleResponce - відповіді, кодуються типом даних SingleResponce,
* respExtensions [1] Extensions

}

ResponcerID = CHOICE {

* byName [1] Name, - або ім’я OCSP сервера
* byKey [2] OCTET STRING – або геш(хеш) від ключа (не довільний геш, а саме SHA-1 від значення ключа)

}

Single Responce = SEQ {

* certI CertID – точна копія такого ж поля з запиту,
* certStatus CertStatus – статус,
* thisUpdate GeneralizedTime - час коли OCSP сервер перевіряв статус,
* nextUpdate [0] GeneralizedTime OPT- до цієї дати статус не зміниться,
* singleExts [1]Extensions OPT

}

Cert Status = CHOICE {

* Good [0] IMPLICIT NULL – статус коли все добре, нічим не кодується,
* revoked [1] IMPLICIT SEQ {  
  revTime GeneralizedTime,  
  revReason [0] CRL Reason OPT}- статус відкликано, кодується тегом 1 і додається інформація – час відкливання і причина відкликання,
* unknown - невідомий статус, кодується тегом 2, наразі тип null

}

Перевіряємо статус - > Перевіряємо час коли було перевірено статус – > потім Перевіряємо відповідь

A picture containing text, blackboard

Description automatically generated

Якщо поле nextUpdate відсутнє можемо запитувати інформацію про оновлення стану в будь-який час.

Розширення відповіді:

1. nonce – якщо клієнт надіслав нонс, то сервер повинен відповісти з нонсем, є ситуації коли сервер все одно може надіслати відповід ез нонса
2. Extended Revoked Definition – якщо сертифікату е існує в центрі сертифікації сервер повинен повертати статус revoked

Розширення елементів:

1. Можна копіювати будь-яке розширення із списку відликаних сертифікатів CRL
2. CRL Reference (URL, CRL Number, час публікації) Сервер може вказати з якого саме списку він брав інформацію про статус

**Спрощений (легкий) протокол OCSP**

Lightweight OCSP RFC 5019

Протокол залишається незмінним, додаються лише додаткові умови до форматів відповідей, запитів, вони спрямовані на пришвидшення роботи та зменшення обєму даних.

Запит:

1. Один сертифікат – один запит – містиь лише один CertId
2. Використовувати лише SHA-1
3. Жодних розширень (окрім можливо нонс(nonce))
4. За модливості не використовувати підписані запити
5. Поле requestor nameтакож не потрібно

Відповідь:

1. Підтримувати лише basic OCSP Responce
2. Відповідь надсилається в будь-якому випадку
3. Якщо респондер не є центром сертифікації OA!=CA, то обовязково вказується сертифікат OA
4. Рекомендовано ідентифікувати респондера за гешем ключа byKey

**Certificate Path Validation – Процедура перевірки ланцюга сертифікатів**

RFC 5280

C1, C2, …. Cn – ланцюг сертифікатів

1. C1 – серифікат точки довіри (trust anchor)
2. Cn – сертифікат який потрібно перевірити
3. Кожен наступий сертифікат повинен бути видавцем наступного

Cj subject = Cj+1 issuer

Схема перевірки

1. Ініціалізація
2. Для усіх i від 1 до n-1
3. Базова перевірка Ci
4. Підготовка до перевірки наступного Ci+1
5. Перевірка Cn
6. Поверенення статусу та додаткова інформація

Вхідні параметри для CPV:

1. Ланцюг сертифікатів
2. Поточний час
3. Політики які прийнятні для користувача
4. Інформація про сертифікат C1 – видавець, ключ, алгоритм
5. Дозвіл на використання відображення політик (policy mappings)
6. Флаг виключності політик
7. Флаг: чи дозволено anyPolicy
8. Обмеження на простір імен: дозволені та виключені. Стосується RDN, emails, Ips

Ініціалізація:

* валідні політики – множина політ в межах яких можно отримати сертифікат
* дерева імен (дозволені, заборонені)
* множина відображення політик
* алгоритм ключа + ключ видавця + імя
* максимальна довжина ланцюга

Базова перевірка сертифіката

1. Перевірка структури, підпису, терміну дії, статусу(для будь-якого механізму)
2. Для несамопідписаних сертифікатів: перевірити обмеження на імена
3. Обробити політики сертифікації

Підготовка до перевірки наступного:

1. Уточнення множини валідних полік (за policy constrains)
2. Зберігається імя + ключ (як видавця наступного сертифікату)
3. Уточнюється древа імен (за name constrains)
4. Перевірка Basic Constraints
5. Перевірка KeyUsage
6. Перевірити усі критичні розширення сертифікату
7. Якщо Ci – не самопідписаний, то max\_path\_len -- (максимальна довжина ланцюга зменшуємо на 1)

Якщо max\_path\_len менше 0, то помилка

1. 3 4) – Якщо pathLenConstraints < max\_path\_len => max\_path\_len = pathLenConstraits

Для останнього сертифікату виконується базова перевірка, далі ми повертаємо множину валідних поліик ,яка сформувалась на основі перевірки. також ми повертаємо ключ, алгоритм ключа, імя сертифікату та ідентифікатор статусу.