Оглавление

[Формат OVAL. Основная логика работы с форматом. 2](#_Toc191964183)

[Объекты, которые были найдены и для чего они используются 4](#_Toc191964184)

[Лишние и обязательные критерии 5](#_Toc191964185)

[Вариант упрощенного формата 7](#_Toc191964186)

[Возможно с JSON будет проще работать, а также парсить и анализировать. 7](#_Toc191964187)

# Формат OVAL. Основная логика работы с форматом.

OVAL (Open Vulnerability and Assessment Language) – это язык, используемый для описания уязвимостей и проверки состояния систем на соответствие определённым условиям.

Формат включает разделы:

* definitions - содержит информацию об уязвимостях, конфигурациях и патчах.
* tests - описывает условия, которые проверяются на системе.
* objects - определяет конкретные элементы системы, на которых проводятся проверки.
* states - задает требуемое состояние объекта.
* variables (в файле не нашел) - определяет внешние переменные.

**definitions -** содержит информацию об уязвимостях, конфигурациях и патчах.

* id — идентификационный номер;
* version — версия;
* class — класс, к которому принадлежит определение.

В теге *metadata*:

* *title* — идентификациz определения пользователем;
* *affected* — информация о системе(ах), для которой было написано определение
* *reference* — связывает определение OVAL с внешней ссылкой. Цель ссылки — связать определение со множеством других источников, посвященных той же проблеме, что и определение OVAL;
* *description* — описание уязвимости;

В теге *criteria*:

описываются, объединенные логическими условиями, критерии и проверки для определения этой уязвимости. Условием являются блоки criteria с параметром логического условия operator, которые могут быть вложенными.

Ссылками на тест является блок *criterion*, указывающий на уникальный элемент типа *test*, содержащийся в данном OVAL-файле.

Tests - описывает условия, которые проверяются на системе.

Тесты возвращают результаты логического типа (true/false).

Главная задача теста — сопоставление текущего состояния объекта с требуемым.

Cложив указанные в критериях условия по логическим цепочкам можно получить необходимые условия для определения наличия уязвимости и выполнения условий определения.

Objects - определяет конкретные элементы системы, на которых проводятся проверки.

States

Компонент *states* задает требуемое состояние объекта. Состояние может быть задано статически или определяться через переменную.

variables

Раздел *variables* определяет внешние переменные, которые можно использовать для настройки элементов из раздела *states*.

# Объекты, которые были найдены и для чего они используются

В каждой уязвимости можно выделить следующие объекты:

1. **Уязвимость RHBA-2019:1992 (cloud-init)**

2. **Уязвимость RHBA-2019:2715 (virt:rhel** **)**

3. **Уязвимость RHBA-2019:3384 (ruby:2.5)**

* **Объекты:**
  + **definition**: описывает уязвимость и её класс (для определения типа проверки).
  + **metadata**: содержит метаданные уязвимости (прим.: заголовок, описание, ссылки на CVE и даты выпуска).
  + **affected**: указывает, какие платформы затронуты (в данном случае Red Hat Enterprise Linux 8).
  + **cve** – описание CVE, CVSS3, CWE.
  + **reference**: ссылки на внешние ресурсы, такие как CVE и баг-трекеры (для дополнительной информации).
  + **criteria**: определяет критерии проверки, которые должны быть выполнены для обнаружения уязвимости.
  + **criterion**: конкретные условия, которые должны быть выполнены.

# Лишние и обязательные критерии

Для выявления каждой уязвимости используются определенные критерии, полагаю, что некоторые критерии можно опустить, т.к. они или дублируют проверку или являются «избыточными».

**В общем случае (не по первым 3-м уязвимостям; предположение)**

**Обязательные критерии:**

1. **Повторяемость** – возможность воспроизвести уязвимость при тех же условиях.
2. **Воздействие на безопасность** – наличие последствий, влияющих на CIA.
3. **Условия возникновения** – описание среды и условий, при которых уязвимость проявляется.

**Лишние критерии:**

1. **Сложность эксплуатации** – не всегда обязательна, т. к. даже сложные уязвимости критичны.
2. **Наличие CVE-идентификатора** – многие уязвимости могут быть новыми и еще не зарегистрированными.
3. **Распространенность** – уязвимость может быть редкой, но опасной.
4. **Наличие автоматизированных инструментов для обнаружения** – важность уязвимости не зависит от наличия сканеров.

**По уязвимостям из файла:**

**Обязательные критерии:**

1. Уязвимость: **RHBA-2019:1992 (cloud-init bug fix)**

#### Обязательные критерии:

**cloud-init версии ниже 0:18.5-1.el8.4**

(test\_ref="oval:com.redhat.rhba:tst:20191992001")

**cloud-init подписан ключом Red Hat redhatrelease2** (test\_ref="oval:com.redhat.rhba:tst:20191992002")

**ОС: Red Hat Enterprise Linux 8 или Red Hat CoreOS 4** (test\_ref="oval:com.redhat.rhba:tst:20191992003", test\_ref="oval:com.redhat.rhba:tst:20191992004")

#### Лишние критерии:

Указание общей информации о системе (Red Hat Enterprise Linux must be installed)

(test\_ref="oval:com.redhat.rhba:tst:20191992005"), т.к. достаточно проверки конкретной версии cloud-init и ОС (не уверен).

2. Уязвимость: **RHBA-2019:2715 (virt:rhel bug fix)**

#### Обязательные критерии:

**qemu-kvm версии ниже 15:2.12.0-65.module+el8.0.0+4084+cceb9f44.5**

(test\_ref="oval:com.redhat.rhba:tst:20192715169")

**qemu-kvm подписан ключом Red Hat redhatrelease2**

(test\_ref="oval:com.redhat.rhba:tst:20192715170")

#### Лишние критерии:

**Проверки множества дополнительных пакетов (libguestfs, sgabios, virt-dib)**

(test\_ref="oval:com.redhat.rhba:tst:20192715173", test\_ref="oval:com.redhat.rhba:tst:20192715181", test\_ref="oval:com.redhat.rhba:tst:20192715189"), т.к. указанные компоненты могут присутствовать в системе, но не обязательно быть связанными с уязвимостью qemu-kvm.

3. Уязвимость: **RHBA-2019:3384 (ruby:2.5 bug fix)**

#### Обязательные критерии:

**Модуль ruby:2.5 включен**

(test\_ref="oval:com.redhat.rhba:tst:20193384065")

**ruby версии ниже 0:2.5.5-105.module+el8.1.0+3656+f80bfa1d**

(test\_ref="oval:com.redhat.rhba:tst:20193384001")

**ruby подписан ключом Red Hat redhatrelease2**

(test\_ref="oval:com.redhat.rhba:tst:20193384002")

#### Лишние критерии:

**Проверки второстепенных зависимостей (rubygem-bundler, rubygem-pg, rubygem-test-unit)**

(test\_ref="oval:com.redhat.rhba:tst:20193384021", test\_ref="oval:com.redhat.rhba:tst:20193384045", test\_ref="oval:com.redhat.rhba:tst:20193384057"), т.к. уязвимость касается основного пакета ruby.

**Возможно, лишний критерий:**

* **Проверка подписи пакета**: если задача — просто определить, установлена ли уязвимая версия, наверное, проверка подписи может быть опущена.

# Вариант упрощенного формата

По упрощенному варианту, есть идеи внести изменения в <definition>, а именно:

1. **Использовать JSON, чтобы** сделать структуру более читаемой (весь файл, не только <definition>).
2. **Сократить вложенные критерии** и оставить только важные проверки.
3. **Группировать данные логически** (например: метаданные, уязвимость, проверка).

**Возможно с JSON будет проще работать, а также парсить и анализировать.**

Структура:

vulnerabilities({  
 "id

"title  
 "severity,  
 "description  
 "references"  
 "criteria"  
})

tests({  
 "id  
 "comment  
 })

objects({  
 "id"  
 "type"  
 "name"  
 })

states({  
 "id"  
 "type"  
 "operation  
 })