Сканер уязвимостей OpenVAS

Поиск уязвимостей при помощи сканера.

[Уязвимости: теория](#_isx9m14oi503)

[Причины возникновения уязвимостей](#_nlmchqxh9i78)

[Bad coding](#_3t52y57c2pj4)

[Архитектурная особенность](#_3vf7movd5fk)

[Ошибки логики работы приложения, компонента или иной сущности](#_7xfqo3scinl6)

[Слабые настройки](#_tb6syzn5htyf)

[Оценка уязвимости](#_1jswz5kervg3)

[Поиск уязвимостей](#_g5dvlz6dctod)

[Поиск уязвимостей по наличию обновлений](#_w9b2znfpbq2b)

[Поиск уязвимости при помощи сканеров](#_v7y2ejgbfwqr)

[Xspider](#_70ueusuh7u7b)

[RedCheck](#_fszfozxsq8sy)

[Nessus](#_a01wxyesih9f)

[OpenVAS](#_m75rdjezppib)

[Scan OVAL](#_eb8451qlebpl)

[OpenVAS](#_kem6dpxru6o6)

[Состав компонентов OpenVAS](#_at1gyh103bmd)

[OpenVAS Scanner](#_r87ljxl0heh9)

[OpenVAS Manager](#_q4r704p4y21t)

[OpenVAS CLI](#_wtt2cq8t2ew)

[Greenbone Security Assistant](#_u6ntb3gk5ccr)

[Правила для обнаружения уязвимостей](#_n6pxyn3ofdvh)

[Установка, настройка и работа с OpenVAS в Kali Linux](#_8ucexlepu5vj)

[Установка OpenVAS в Kali Linux](#_3znfkncdxa5l)

[Проверка работы сервисов OpenVAS](#_ybusq0felnkv)

[Добавление и смена пароля пользователя](#_a4ug91ww0uu7)

[Практическая часть](#_2xahui9zlgvz)

[Задание 1. Быстрое сканирование (immediate scan)](#_jupea4mb2hig)

[Задание 2. Задать цель вручную и провести сканирование цели](#_vqxvx1xisblm)

[Задание 3. Создать конфиг для сканирования узла с ОС Debian Linux и просканировать узел](#_rwk2l0bllubf)

[Выводы](#_rir8un7orau7)

Практическое [задание](#_j0omyxcerqv1)

[Дополнительные материалы](#_30j0zll)

[Используемая литература](#_1fob9te)

# Уязвимости: теория

Определимся с терминологией. Согласно ISO 27000:

* **уязвимость** (**vulnerability**) — это слабое место актива или меры и средства контроля и управления, которое может использовать угроза;
* **актив (asset)** — то, что имеет ценность для организации;
* **мера и средство контроля и управления (control)** — предназначены для менеджмента риска. Включает политики, процедуры, рекомендации, практические приемы или организационные структуры (административные, технические, управленческие или правовые).

Под уязвимостью будем понимать слабость, эксплуатируя которую, можно реализовать угрозу.

Уязвимости можно разделить на те, что присутствуют в программных и технических компонентах, и те, которые появились вследствие несоблюдения процедур безопасности.

С каждой уязвимостью будет связана угроза: если она реализуется в системе, будут последствия.

Чем серьезнее последствия, тем выше показатель угрозы и уязвимости, приводящей к ее реализации, при их оценке. Но надо учитывать, что за угрозой могут стоять нескольких уязвимостей.

Рассмотрим, каким образом возникают уязвимости и происходит их оценка.

## Причины возникновения уязвимостей

### Bad coding

Обычная причина уязвимостей в ПО — ошибки, которые возникают при написании программ. Наиболее типичный пример — уязвимость **buffer overflow**. В «Википедии» о ней пишут так: *«Записывая данные в буфер, можно осуществить запись за его границами и изменить находящиеся там данные (в частности, адрес возврата). Если программа имеет особые привилегии — например, запущена с правами root, — злоумышленник может заменить адрес возврата на адрес шелл-кода. Это позволит ему исполнять команды в атакуемой системе с повышенными привилегиями»*.

Подобная процедура возможна в том числе потому, что в языках программирования нет встроенной защиты от переполнения буфера. Особенно это относится к C/C++, исключение — JavaScript.

### Архитектурная особенность

Часто уязвимость присутствует на аппаратном уровне: в процессоре или ОЗУ. Уязвимость **Meltdown** использует ошибку реализации спекулятивного выполнения команд в процессорах Intel и ARM. Из-за нее при спекулятивном выполнении инструкций чтения из памяти процессор игнорирует права доступа к страницам. При запуске специальной программы уязвимость позволяет локальному атакующему несанкционированно читать привилегированную память, которую использует ядро ОС.

Рассмотрим еще уязвимость **Rowhammer**, которой подвержены некоторые чипы **DDR3**. Злоумышленник может запустить программу, которая будет тысячи раз за долю секунды обращаться к конкретным рядам участков в модуле памяти. Она «постукивает по ним молотком» (hammering), пока электромагнитное излучение не проникает в соседний участок памяти. Это может изменить значения битов в нем с нулей на единицы и наоборот.

### Ошибки логики работы приложения, компонента или иной сущности

Иногда при разработке приложения авторы не учитывают особенности работы с одним из компонентов. Как в случае с уязвимостью **Sql injection**, при которой в имеющейся SQL-запрос добавляются конструкции, существенно изменяющие его логику работы. Такое нарушение в web-приложении может привести к уязвимостям: **XSS**, **CSRF**, **Clickjacking** и другим.

### Слабые настройки

Как правило, это стандартные учетные данные оборудования (логин, пароль) или те, что легко подобрать. Часто в устройстве есть предустановленная учетная запись сервиса, доступного из глобальной сети (например, **telnet**). Возможны две ситуации:

* предустановленные учетные данные — актуально для устройств сегмента **IoT**;
* слабые учетные данные — которые пользователь установил сам. Актуально всегда.

Важно не использовать подобные данные для аутентификации привилегированных пользователей.

## Оценка уязвимости

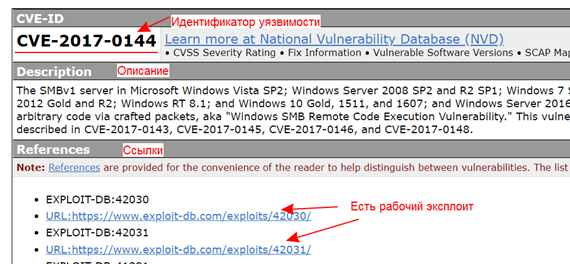
Каждая уязвимость в технических или программных решениях имеет свой рейтинг и идентификатор.

Общепринятый стандарт описания уязвимости — формат **CVE** (**Common Vulnerabilities and Exposures**):

**CVE — год — номер**

Пример: **CVE-2017-0144.**

CVE поддерживает организация [MITRE](https://cve.mitre.org/). На ее сайте можно посмотреть описание уязвимости и связанную с ней информацию:



Особо опасным уязвимостям присваиваются имена: уязвимость под номером CVE-2017-5754 — это Meltdown.

Чтобы оценивать уровень опасности, применяется стандарт CVSS. Каждой уязвимости присваивается вектор: для уязвимости CVE-2014-2363 это **AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H**. Также для уязвимости определяется уровень от 0 до 10. Чем выше этот показатель, тем опаснее уязвимость.

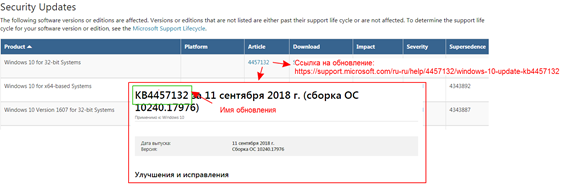
Используется две версии CVSS: 3 и 2. На рисунке — описание уязвимости CVE-2018-8349 с сайта <https://nvd.nist.gov>:



Дополнительно информацию об уязвимостях можно посмотреть на следующих ресурсах:

* [National Vulnerability Database](https://nvd.nist.gov);
* [Vulners.com](https://vulners.com);
* [БД уязвимостей ФСТЭК](http://www.bdu.fstec.ru).

Информация об уязвимостях агрегируется и в описании обновлений на сайтах производителей ОС, где они были обнаружены. Обновление как мера для закрытия уязвимости — наиболее типичное решение. Например, уязвимость CVE-2018-8475 описывается на [странице Security update guide портала Microsoft](https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-8475). Здесь же можно просмотреть обновление, которое «закрывает» данную уязвимость:



Следует понимать, что если уязвимость — результат действий пользователя, то в базу данных CVE она обычно не попадает. Например, если установили пароль «admin» при входе в систему.

# Поиск уязвимостей

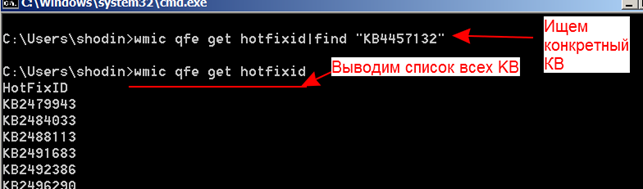
## Поиск уязвимостей по наличию обновлений

Самый простой способ обнаружить, подвержена система уязвимостям, — проверить, установлены ли обновления, которые закрывают их. Если нет — возможно, система уязвима. В Windows это можно выполнить командой:

|  |
| --- |
| wmic qfe get hotfixid |

В этом случае выводится список всех установленных обновлений. Искать в этом перечне можно с помощью этой команды:

|  |
| --- |
| wmic qfe get hotfixid|find "KB4457132" |



В системах на базе Debian Linux можно вывести список возможных обновлений и отфильтровать в них вывод по ключевому слову «security»:

|  |
| --- |
| sudo apt list --upgradable|grep security |

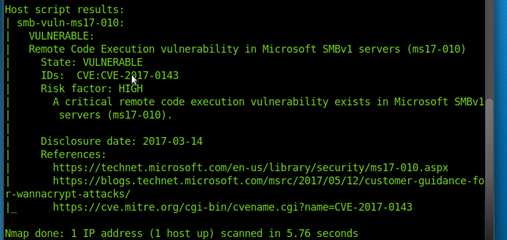
Чтобы искать уязвимость эти способом, нужно точно знать, какое обновление должно отсутствовать.

Если уязвимость серьезная, для нее может существовать плагин для сканера **nmap**. В таком случае команда следующая:

|  |
| --- |
| nmap --script smb-vuln-ms17-010.nse 192.168.56.1 |

* **--script smb-vuln-ms17-010.nse** — это имя сценария для тестирования на уязвимости;
* **192.168.56.1** — адрес тестируемого узла.

Резул



Полный список сценариев, эксплуатирующих различные уязвимости, можно посмотреть [на сайте **nmap.**](https://nmap.org/nsedoc/categories/vuln.html) Но не все уязвимости представлены в виде плагинов для сканера.

Гораздо практичнее искать уязвимости с помощью специальных программ-сканеров.

## Поиск уязвимости при помощи сканеров

Как правило, сканер содержит сценарии для поиска уязвимостей и движок, который проверяет по сценариям конечные узлы сети или локальный компьютер, если сканер не сетевой.

Стандарт при описании уязвимости — **OVAL** (**Open Vulnerability and Assessment Language**), открытый язык описания и оценки уязвимостей. Он позволяет использовать один XML-файл, в котором уже описана уязвимость, метод ее определения, ссылка на патч, на CVE и многое другое. Этот формат стандартный, чтобы принимать информацию для сканеров безопасности. Официальный репозиторий правил можно [скачать сайте OVAL](http://oval.mitre.org/repository/).

Рассмотрим популярные сканеры и их функциональные возможности.

### **Xspider**

Разработка компании Positive Technologies с лицензией ФСТЭК. Особенности:

* контролирует изменения на сканируемых узлах и позволяет получить полную картину защищенности в динамике;
* проверяет слабость парольной защиты: выполняет оптимизированный подбор паролей практически во всех сервисах, требующих аутентификации;
* глубоко анализирует контент веб-сайтов, выявляет уязвимости в скриптах (SQLi и XSS, запуск произвольных программ);
* анализирует структуру HTTP-серверов, чтобы искать слабые места в конфигурации;
* выполняет расширенную проверку узлов под управлением Windows;
* проводит проверки на нестандартные DoS-атаки.

**Xspider** — платное решение.

### RedCheck

Разработка компании АЛТЭКС-СОФТ, лицензированная ФСТЭК. Особенности:

* ищет уязвимости;
* ищет и устанавливает обновления;
* проводит инвентаризацию сети;
* контролирует целостность исполняемых файлов и служебных библиотек;
* сканирует сеть;
* подбирает пароли.

Это платный продукт.

### Nessus

Один из первых сканеров уязвимостей. Особенности:

* интегрируется с Metasploit;
* есть бесплатная версия для домашней сети и платная для корпоративной;
* экспортирует отчеты в различных форматах.

Платный продукт.

### OpenVAS

Большинство компонентов в **OpenVAS** — по лицензии GNU GPL, интегрирован в Kali Linux. Особенности:

* интегрируется с Metasploit;
* экспортирует отчеты в различных форматах;
* дает рекомендации, как устранить уязвимость;
* предустановлен в Kali linux.

Потребляете много ресурсов.

### Scan OVAL

**Scan OVAL** — утилита ФСТЭК для тестирования локального ПК на уязвимости. Особенности:

* сканирует ПК по базе данных OVAL;
* экспортирует отчеты.

Это пробная версия, сканирует только локальный ПК.

Практические рекомендации по выбору сканеров уязвимостей:

1. Сформулируйте цель сканирования — пентест или администрирование сети. Ряд сканеров оптимизированы именно для пентеста, и они, как правило, предустановлены в соответствующие ОС.
2. Определите, нужен ли сканер уязвимостей, лицензированный ФСТЭК. Это полезно, если сканирование на уязвимости будет частью процедуры, которая обеспечивает защиту конфиденциальной информации в организации (например, персональных данных).
3. Определите, как будет обрабатываться отчет об обнаруженных уязвимостях и надо ли его сохранять. Многие сканеры поддерживают экспорт результатов проверки в файл.
4. Решите, будут ли сканироваться хосты, сети и сетевые устройства. В некоторых случаях достаточно просканировать локальный хост, запустив на нем программу.

# OpenVAS

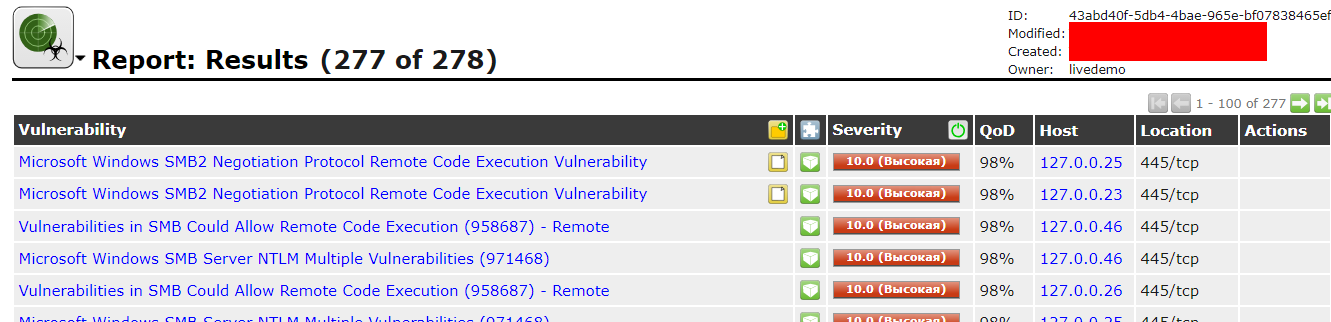
OpenVAS — это набор решений для комплексного сканирования сетевых ресурсов на уязвимости и управления найденными.

При помощи OpenVAS можно:

* искать известные уязвимости — с уже рассчитанным рейтингом CVSS по известным CVE — и получать рекомендации по их устранению;
* искать свежие уязвимости при периодических проверках;
* планировать проверки — например, проверять сервер в нерабочее время;
* создавать отчеты о результатах проверок — чтобы обосновать внедрение механизмов защиты;
* получать рекомендации по внедрению механизмов защиты.

OpenVAS — это в первую очередь сканер, который работает по правилам. Если для уязвимости нет правил NVT в базе OpenVAS, сканер ее обнаружить не сможет.

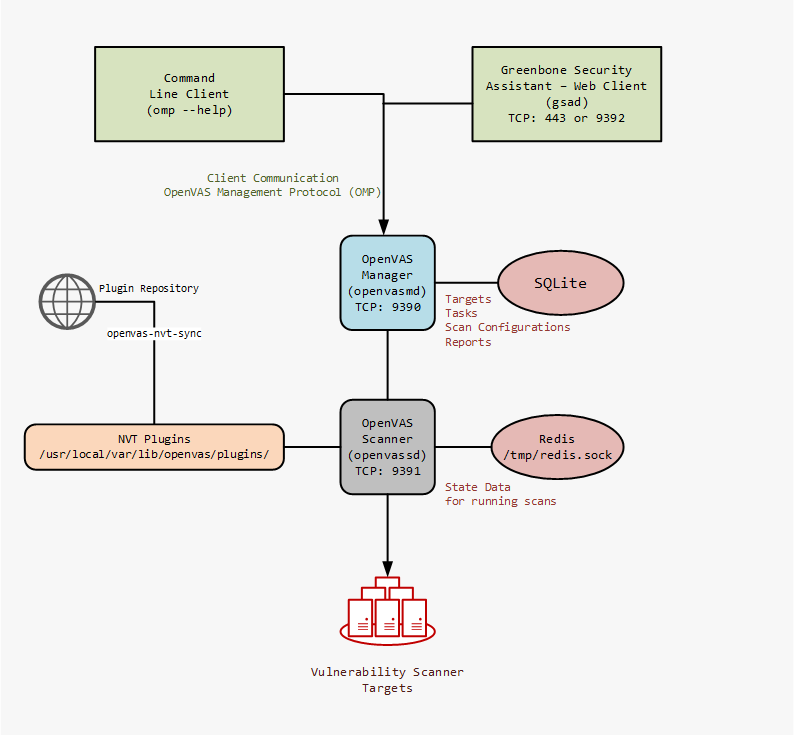
Пример отчета об обнаружении уязвимостей:



# Состав компонентов OpenVAS

* **OpenVAS Scanner**;
* **OpenVAS Manager**;
* **OpenVAS CLI**;
* интерфейс **Greenbone security assistant**.

Взаимосвязи компонентов — [иллюстрация с сайта hackertarget](https://hackertarget.com/openvas-tutorial-tips/):



## OpenVAS Scanner

Сервис для сканирования компонентов сети. Сканер ищет известные уязвимости по базе Network Vulnerability Tests (NVT), которая представляет обобщенную информацию о них. NVT можно считать правилами поиска уязвимостей. Основа работы сканера — конфигурационный файл **openvassd.conf**, который обычно расположен в каталоге **/usr/local/etc/openvas/**.

Управляют OpenVAS Scanner при помощи команды **openvassd**.

## OpenVAS Manager

Это основной сервис, который управляет сканером и позволяет сохранять результаты в sqlite-подобную БД. Обмениваться информацией между сканером и менеджером помогает специальный протокол **omp**. **OpenVAS Manager** выступает как прослойка между **OpenVAS Scanner** и его клиентами.

Управлять **OpenVAS Manager** можно при помощи команды **openvasmd**.

## OpenVAS CLI

Чтобы управлять этими компонентами, используется **OpenVAS CLI**. Это командная оболочка для работы сервисов OpenVAS.

OpenVAS можно управлять и из консоли: создавать задачи для сканирования, загружать отчеты о его результатах.

Протоколом **omp** управляют при помощи команды **omp**.

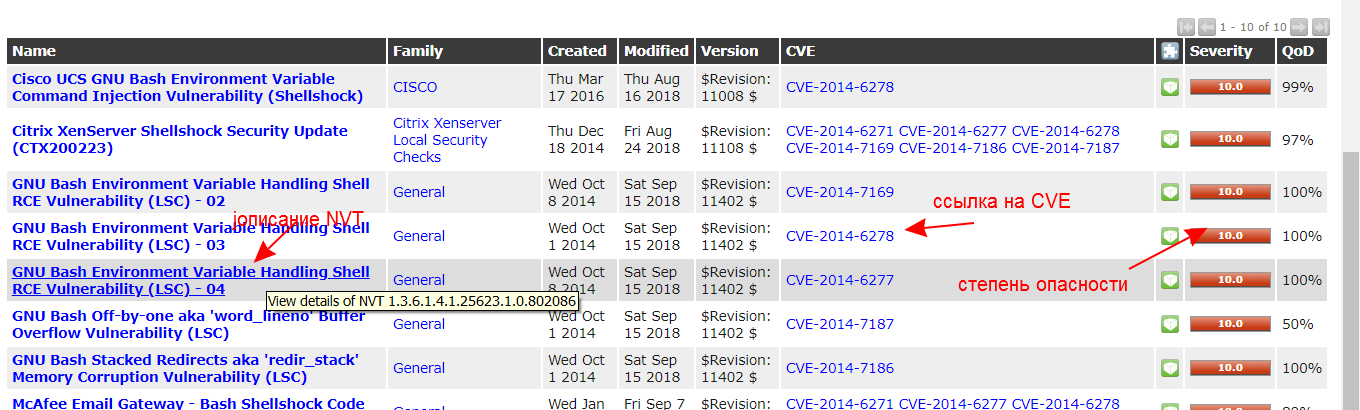
## Greenbone Security Assistant

Чтобы работать с OpenVAS было проще, используется web-интерфейс Greenbone Security Assistant. Это надстройка над **OpenVAS cli**, управлять которой можно через команду **gsad**.

# Правила для обнаружения уязвимостей

Чтобы обнаружить уязвимости, используются Network Vulnerabilities Tests (далее NVT). Это плагины — файлы вида **\*.nasl**, — написанные при помощи языка **nasl** (**Nessus Attack Scripting Language**). Каждый NVT связан с уязвимостью, а у нее есть показатель рейтинга Severity. Он говорит об опасности угрозы, которую можно реализовать при помощи уязвимости.

Для одного из видов Shellshock, уязвимости CVE-2014-6278, есть NVT, который позволяет находить ее и связан с рейтингом CVSS для нее:



Каждый NVT содержит следующую информацию (наиболее важные данные выделены жирным шрифтом):

1. **Наименование**. Обычно указывается имя NVT, например “GNU Bash Environment Variable Handling Shell RCE Vulnerability (LSC) – 04”.
2. Время создания: Wed Oct 8 06:41:49 2014.
3. Время изменения: Sat Sep 15 09:13:36 2018. Периодически параметры NVT переполняются и меняется показатель Severity для конкретного NVT.
4. **Резюме (Summary)** — описание уязвимости, с которой связан конкретный NVT: “This host is installed with GNU Bash Shell and is prone to remote command execution vulnerability”.
5. Подверженное уязвимости ПО (Affected Software/OS): GNU Bash through 4.3 bash43-026.
6. **Количественная оценка уязвимости (Vulnerability Scoring).** Как правило, указывается в виде параметра CVSS: “CVSS base: 10.0. CVSS base vector: AV:N/AC:L/Au:N/C:C/I:C/A:C”. Чем выше оценка уязвимости по CVSS, тем она опаснее.
7. **Суть уязвимости (Vulnerability Insight)**. Описание уязвимости и ее механизмов: “GNU bash contains a flaw that is triggered when evaluating environment variables passed from another environment. After processing a function definition, bash continues to process trailing strings. Incomplete fix to CVE-2014-7169, CVE-2014-6271”.
8. **Метод определения уязвимости** **(Vulnerability Detection Method)**. Описание метода определения уязвимости: “Login to the target machine with ssh credentials and check its possible to execute the commands via GNU bash shell”. Также указывается параметр качества воспроизведения уязвимости: “Quality of Detection: exploit (100 %)”.
9. **Влияние уязвимости** **(Impact)**. Описание последствий от реализации уязвимости злоумышленником: “Successful exploitation will allow remote or local attackers to inject shell commands, allowing local privilege escalation or remote command execution depending on the application vector”.
10. **Возможное решение проблемы (Solution)**. Возможные решения для найденной уязвимости.
11. Список справочных ссылок (CVE, Bugtraq ID, уведомления CERT), ссылки на заметки и переопределения, связанные с данным NVT. В разделе собрана полезная информация об уязвимости.

OpenVAS предлагает следующие возможные решения для проблем:

1. **Workaround** — есть описание конфигурации или сценария специальной установки, которое можно использовать, чтобы прикрыть уязвимость.
2. **Mitigation** — есть описание конфигурации или сценария специальной установки, которое можно использовать, чтобы снизить риск от эксплуатации уязвимости. Полностью не решает проблему с уязвимостью в затронутых продуктах.
3. **VendorFix** — есть официальное исправление от производителя затронутого продукта. Если нет специальных указаний, подразумевается, что исправление полностью устраняет уязвимость.
4. **NoneAvailable** — решение проблемы пока отсутствует. В сообщении должно быть объяснено, почему решения нет.
5. **WillNotFix** — исправления для уязвимости нет и не предвидится. Так бывает с устаревшими продуктами. В сообщении должно быть объяснено, почему решения нет в конкретном случае.

Чем полнее и актуальнее база NVT для OpenVAS, тем точнее результаты сканирования.

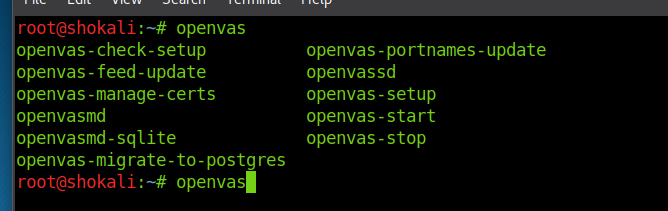
# Установка, настройка и администрирование OpenVAS в Kali Linux

## Установка OpenVAS в Kali Linux

OpenVAS устанавливается в Kali Linux при помощи этой команды:

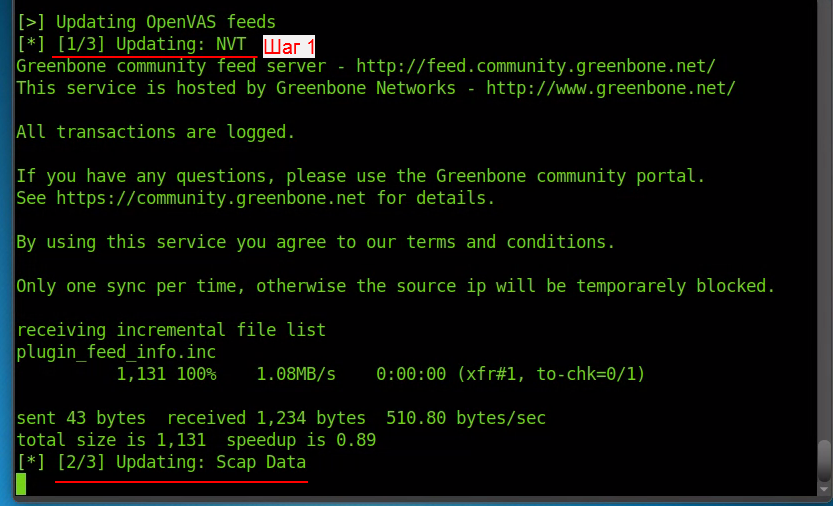
|  |
| --- |
| apt-get install openvas |

В системе появятся эти компоненты:

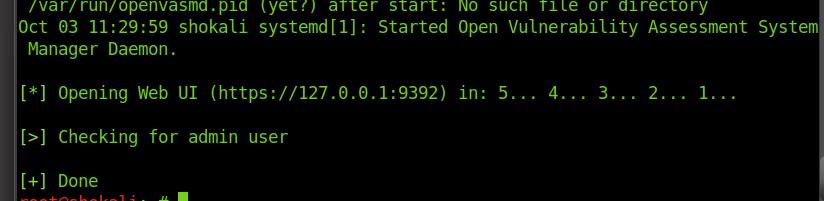


Далее необходимо выполнить команду **openvas-setup**. Установка проходит в три этапа:

* **Updating NVT** — обновление базы NVT;
* **Updating SCAP data** — обновление базы SCAP, которая содержит БД автоматизированного управления уязвимостями OpenSCAP (Security Content Automation Protocol);
* **Updating CERT data** — обновление сертификатов.



В результате команды **openvas-setup** обновится база **NVT**, **scap** и сертификатов, перезапустится OpenVAS.



При установке будет создан пользователь admin, пароль которого надо зафиксировать, так как он генерируется автоматически. В дальнейшем его можно будет изменить.

Потом автоматически откроется браузер со страницей входа в интерфейс Greenbone Security Assistance. Здесь надо ввести пароль пользователя **admin**, сгенерированный ранее. В дальнейшем запускать систему можно командой **openvas-start.**

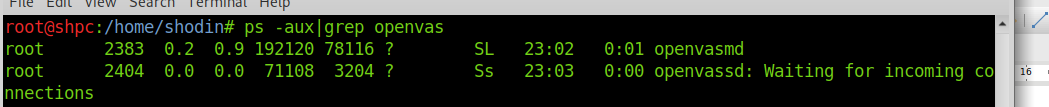
## Проверка работы сервисов OpenVAS

Для проверки необходимо проконтролировать запуск необходимых компонентов:

* OpenVAS Manager;
* интерфейс Greenbone security assistant;
* OpenVAS Scanner (опционально).

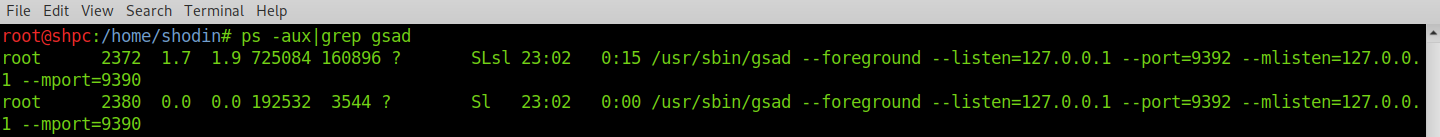
Запуск сервисов **OpenVAS manager** и **OpenVAS Scanner** можно проверить при помощи команды:

|  |
| --- |
| ps -aux|grep openvas |



Запуск сеcleaрвиса **Greenbone Security Assistant** можно проверить при помощи команды:

|  |
| --- |
| ps -aux|grep gsad |



Если в системе запущен только OpenVAS и других сетевых сервисов нет, можно просмотреть активные сетевые соединения, которые используются компонентами. Поможет эта команда:

|  |
| --- |
| netstat -antp |

щзут

Надо проконтролировать, что запущен **openvasmd** и интерфейс **Greenbone**. В примере интерфейс Greenbone запущен на http- и https-порту.

## Добавление и смена пароля пользователя

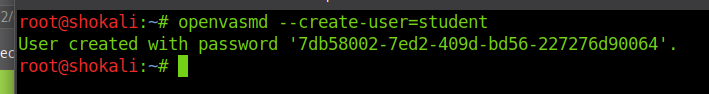
Сначала надо вывести список всех пользователей, чтобы узнать пользователей Openvas:

|  |
| --- |
| openvasmd –get-users |



Чтобы создать нового пользователя, используется эта команда:

|  |
| --- |
| openvasmd --create-user=student |



Сменить пароль у пользователя можно при помощи такой команды:

|  |
| --- |
| openvasmd --user=student --new-password=admin123. |

После этого необходимо перезапустить **openvas** командами **openvas-stop** и **openvas-start.** Теперь можно войти в систему с новым паролем.

# Практическая часть

Сканируя систему, злоумышленника ищет такие уязвимости, эксплуатация которых позволит ему обойти механизмы безопасности и развить атаку. Он может проводить:

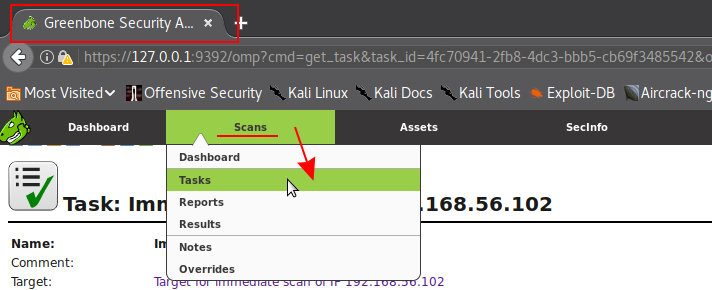
* **Сканирование, которому не задаются параметры.** Цель — определить сущности, которые можно использовать, чтобы проникнуть на атакуемый узел. Как правило, при таком виде сканирования собираются сведения для более глубокого исследования: достаточно добыть информацию о версиях ОС. Или извлекается максимум информации о всех уязвимостях, если время исследования некритично.
* **Глубокое сканирование.** При этом злоумышленнику известны параметры сканирования: например, тип и версия ОС исследуемого хоста. Задача при организации такого сканирования — поиск конкретных уязвимостей (например, присущих определенной ОС).

Рассмотрим способы организации этих видов сканирования.

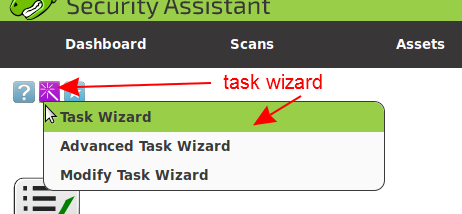
## Задание 1. Быстрое сканирование (immediate scan)

Обычно об исследуемом узле не известно ничего, кроме его имени или адреса. Поэтому в OpenVAS есть мастер сканирования, при помощи которого можно быстро организовать исследование узла или их группы, не думая о настройках. Для этого необходимо:

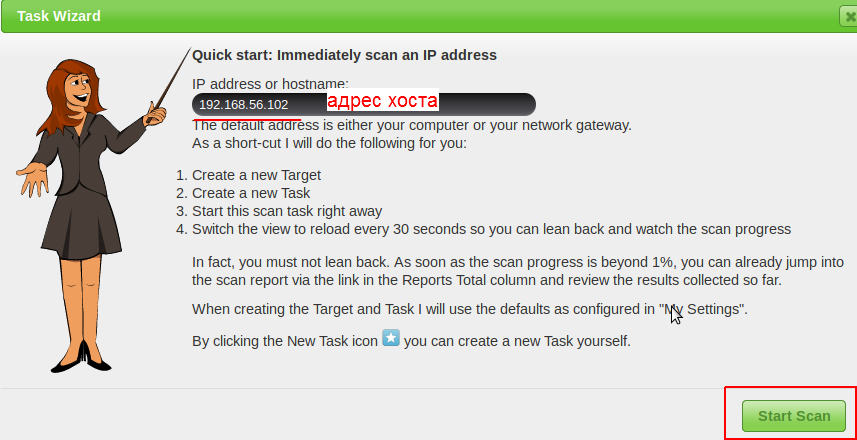
Перейти в раздел **Scans — Tasks** в интерфейсе Greenbone Security Assistance.



В нем запустить мастер создания задач (**Task Wizard**):

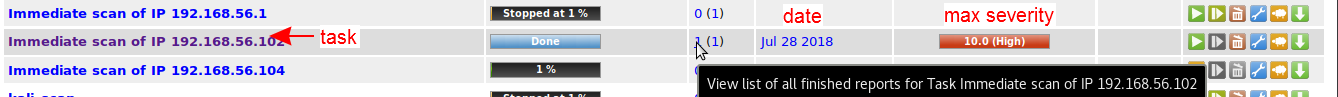


В открывшемся окне вводим параметры хоста и нажимаем кнопку **Start Scan.** После этого будет автоматически выполнено четыре действия, изложенных на рисунке ниже:

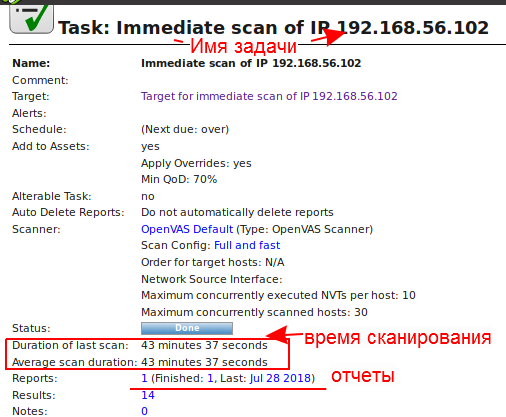


Остается только дождаться результатов сканирования. При данном способе сканирования задача создается с профилем **Full and fas**t, в котором содержатся 47556 правил NVT для сканирования.

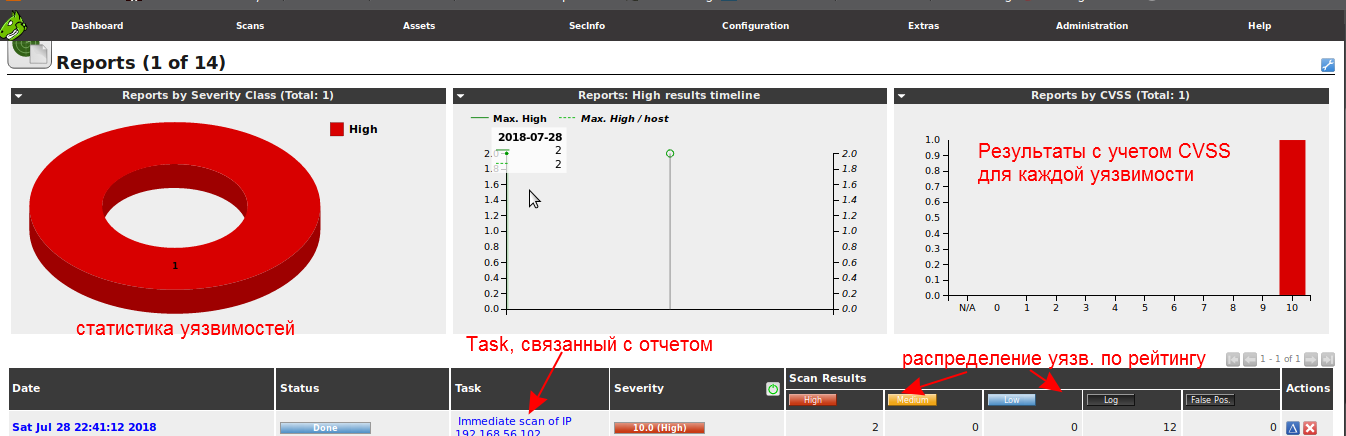
Сканирование может быть долгим. По его завершении можно увидеть максимальный рейтинг Severity для найденных уязвимостей, открыть задачу и просмотреть подробности:



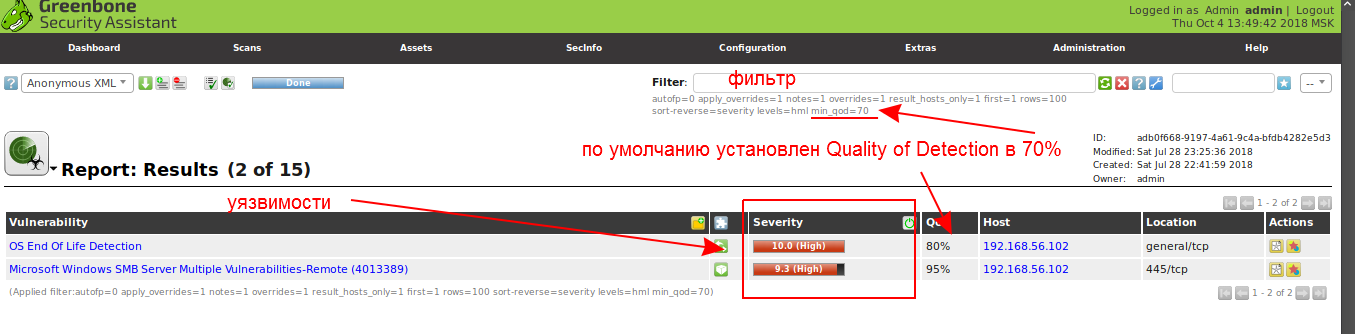
В подробной сводке указано общее время сканирование и предоставлены отчеты для просмотра:



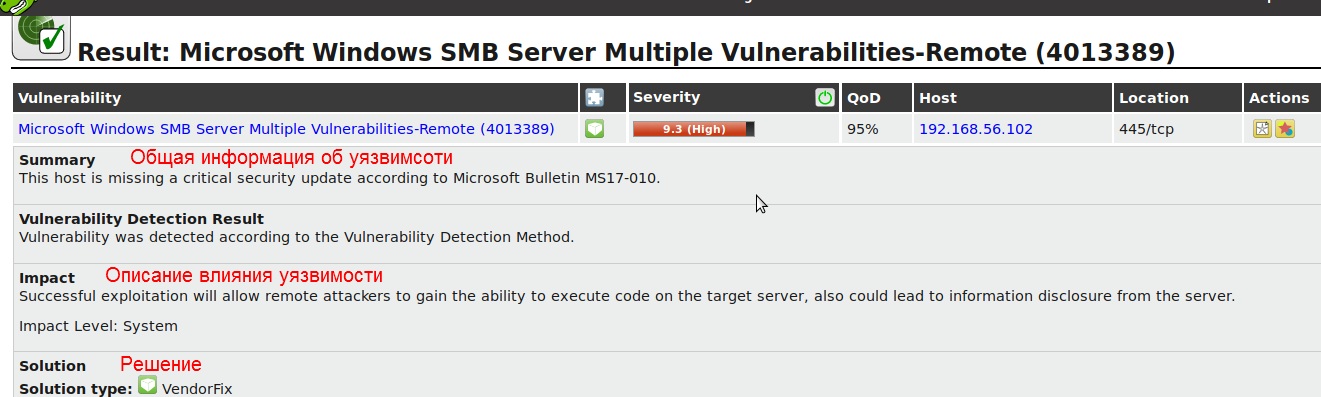
Далее по ссылке из раздела Reports можно просмотреть отчет с подробной информацией об обнаруженных уязвимостях:



По ссылке из раздела Date расположена более подробная сводка. По умолчанию отображаются уязвимости с параметром **Quality of detection** не ниже 70 %. Это можно убрать в настройках фильтра:



Чтобы ознакомиться с информацией о найденной уязвимости, надо перейти по ссылке с ее именем. В открывшемся окне будет описание уязвимости и решение, которое предлагает OpenVAS:

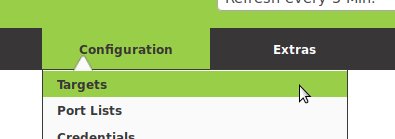


## Задание 2. Задать цель вручную и провести сканирование цели

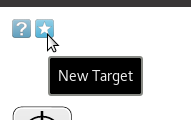
Заранее настроить цель сканирования полезно, если:

* о цели сканирования уже собрана информация, и это может сократить время процедуры;
* необходимо задать параметры сканирования — например, данные для подключения к хосту;
* при планировании периодических проверок ресурсов одного и того же хоста на наличие в нем уязвимостей.

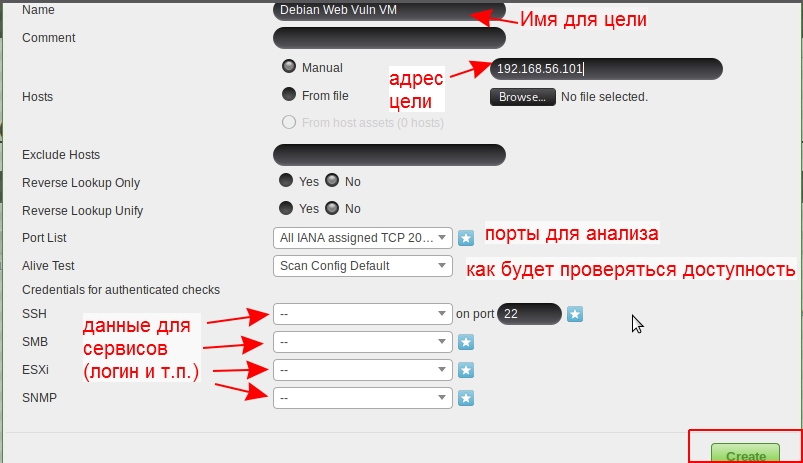
Создадим новую цель для сканирования. Переходим в раздел **Configuration — Targets:**



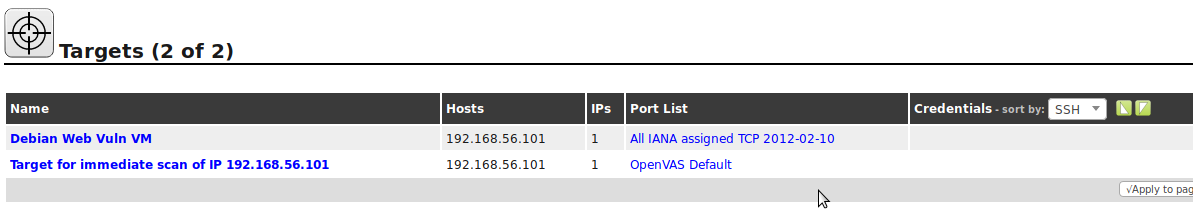
Выбираем пиктограмму New Target:



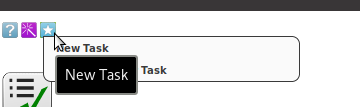
Цель можно задать списком адресов, указать параметры сервисов, которые будем использовать при сканировании (логин/пароль для SSH).



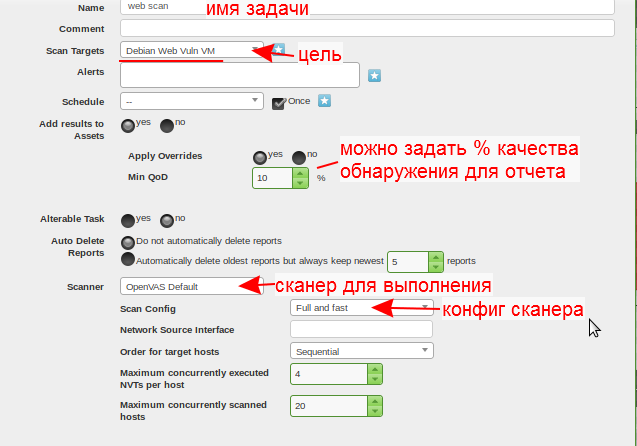
Цель добавится в список:



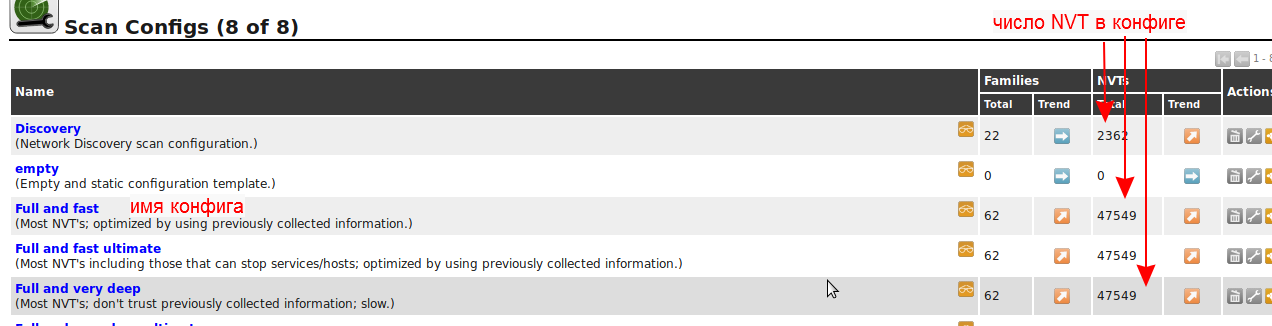
Когда добавили цель, можно создать новую задачу для сканирования через меню **Scans — Tasks**:



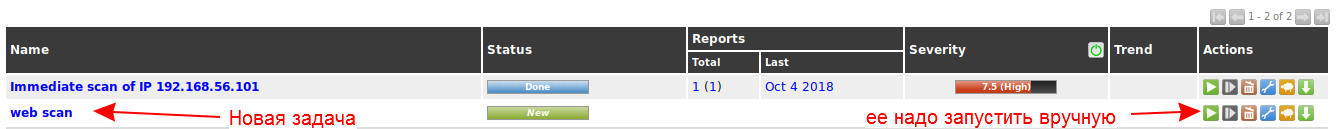
Когда добавляем задачи, выбираем созданную ранее цель. Цели подставляются из меню **Targets.** Для сканирования используются две важные настройки: сканер и его конфигурация. При создании задачи добавляется параметр QoD (в %) как фильтр для отчета. В дальнейшем фильтр можно изменить.



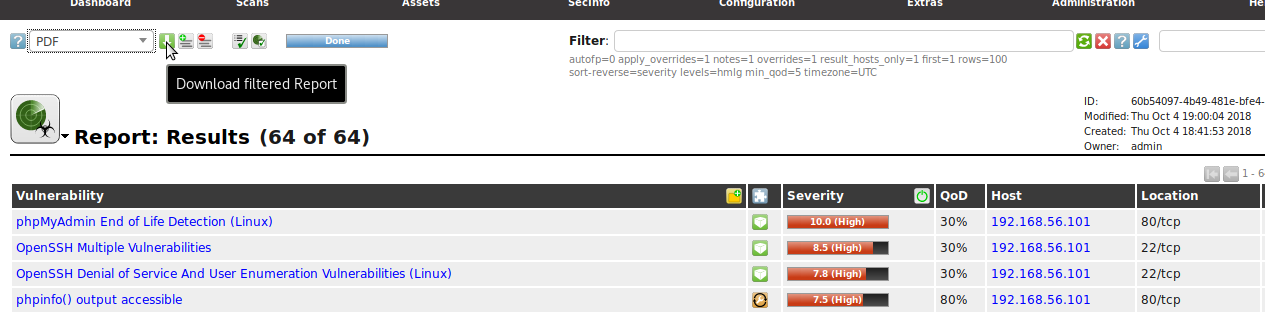
Когда настраиваем конфигурацию сканирования, значения берем из меню **Configuration — Scan config**. Все конфигурации, которые можно создавать самостоятельно, отличаются по количеству NVT. Их число можно изменять при создании нового конфига, так как оно прямо влияет на время сканирования.



Созданную задачу можно запустить и затем проанализировать результат.



Отметим, что отчет о сканировании с учетом примененных фильтров, например QoD <= 5 %, можно выгрузить в нескольких форматах, в том числе PDF.



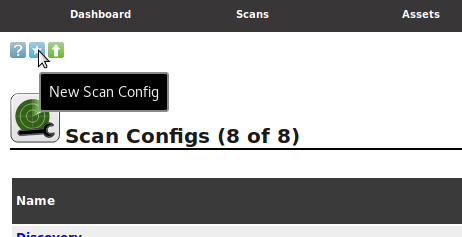
## Задание 3. Создать конфиг для сканирования узла с ОС Debian Linux и просканировать узел

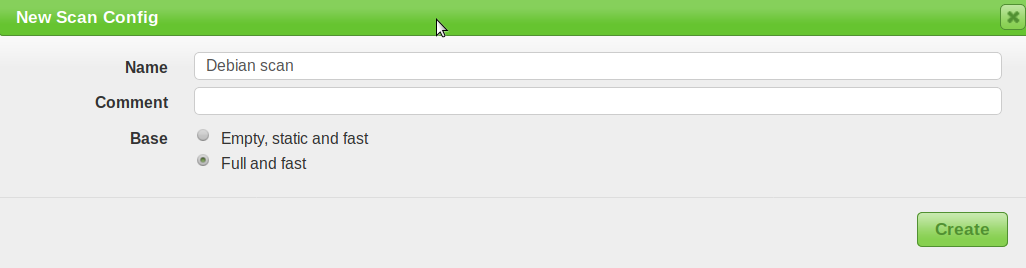
Предположим, что точно знаем тип ОС на хосте и данные для подключения к нему. В этом случае можно провести более глубокое сканирование узла, которое позволит выявить уязвимости, недоступные для внешнего сканирования. Это полезно, чтобы:

* организовать периодические проверки хоста на уязвимости;
* задать правила для поиска уязвимостей в конкретной ОС. Это сократит время сканирования.

Недостаток данного метода в том, что надо знать о хосте дополнительную информацию — как минимум тип ОС.

Создадим новый конфиг в меню **Configuration — Scan configs:**



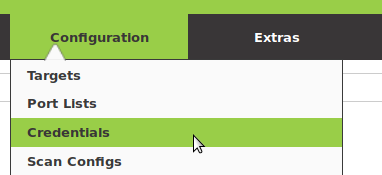


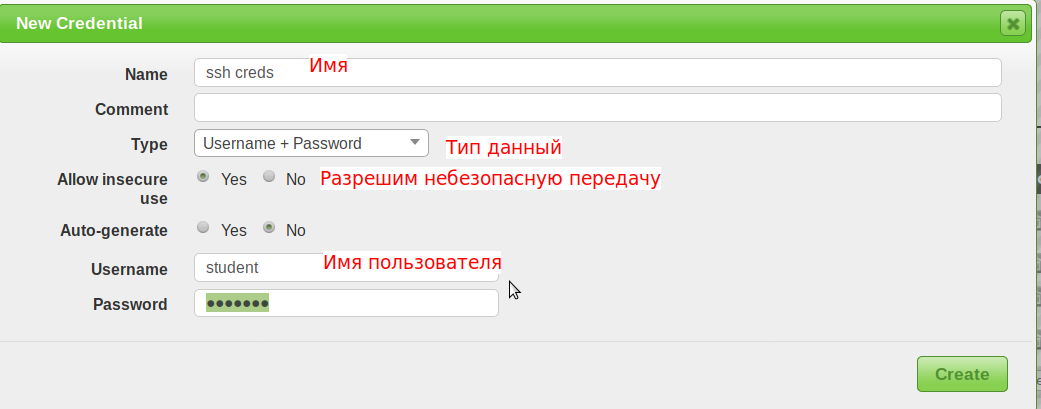
Зададим имя и шаблон конфига для основы. После этого конфиг добавится в список. Нужно прикрепить к нему NVT, которые отвечают за сканирование Debian, и убрать лишние NVT, сняв флажки. Чтобы проверить Debian, не нужны NVT для других ОС.

Из раздела **Port scan** добавим два компонента: **Nmap** (**NASL** **wrapper**) и **Ping** **host**. Созданный набор сохраняем.



Чтобы сканировать ОС на наличие уязвимостей или неустановленных обновлений, рекомендуем задать данные для подключения — например, по **SMB** или **SSH**. Для этого переходим в раздел **Configuration — Credentials.**

Создадим новый набор данных для подключения:



Добавим новую задачу для сканирования (**Scans — Task — Advanced Task Wizard**) и прикрепим к ней конфиг и набор данных для подключения к SSH:



В дальнейшем можно применять этот набор данных для подключения не только для SSH. Созданные через wizard задачи запускаются автоматически, остается дождаться результатов сканирования.



# Выводы

Мы рассмотрели базовые особенности сканера OpenVAS:

* изучили состав и структуру компонентов OpenVAS;
* узнали, как просканировать ОС на уязвимости;
* создали конфигурации для сканирования, при помощи которых можно расширять и сужать круг уязвимостей для обнаружения;
* научились использовать регистрационные данные, чтобы подключаться к удаленной системе и расширять зону покрытия сканирования.

Сканер можно применять шире:

* установить OpenVAS в ОС Linux, отличную от Kali Linux, и использовать его как отдельное решение для сканирования сетевых ресурсов;
* планирования с помощью OpenVAS стратегию внедрения политики безопасности и приоритизации рисков;
* использовать OpenVAS в сочетании с **metasploit**, если требуется единая платформа для анализа защищенности сети.

# Практическое задание

1. Установить OpenVAS в Kali Linux.
2. Установить систему DVL Linux в качестве виртуальной машины (ссылка для скачивания <https://sourceforge.net/projects/virtualhacking/files/os/dvl/DVL_1.5_Infectious_Disease.iso/download>, можно просто загрузиться с iso образа), настроить сетевой доступ к ней со стороны Kali Linux и просканировать систему DVL Linux на наличие уязвимостей. Формат сдачи - отчет OpenVAS в формате pdf или html.
3. \* Установить виртуальную машину на базе Windows 7 (8, 8.1 или 10), активировать сетевой доступ к общим папкам. Просканировать ВМ при помощи OpenVAS с использованием данных протокола SMB. Формат сдачи - отчет OpenVAS в формате pdf или html.

Задание со звездочкой\* — повышенной сложности.

# Дополнительные материалы

1. [Описание Openvas CLI](http://www.openvas.org/src-doc/openvas-cli/index.html).
2. [Советы по настройке и использованию OpenVAS из командной строки](https://hackertarget.com/openvas-tutorial-tips).
3. [Развертывание OpenVAS](https://www.hackingtutorials.org/scanning-tutorials/vulnerability-scanning-openvas-9-pt-1).
4. [Сканирование сети (а не отдельного узла) в OpenVAS](https://www.hackingtutorials.org/scanning-tutorials/vulnerability-scanning-with-openvas-9-scanning-the-network).
5. [О развертывании и использовании OpenVAS в Windows-подобной среде](https://myworldofit.net/?series=a-windows-sysadmin-installs-and-uses-openvas).
6. [Как просканировать узел на наличие уязвимости к WannaCry](https://www.cb-net.co.uk/microsoft-articles/vulnerability-scanning-for-ms17-010-4013389-using-openvas-in-a-docker-container/).

# Используемая литература

1. [https://ru.wikipedia.org/wiki/Переполнение\_буфера.](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B1%D1%83%D1%84%D0%B5%D1%80%D0%B0)
2. [https://www.veracode.com/security/buffer-overflow.](https://www.veracode.com/security/buffer-overflow)
3. [https://habr.com/company/1cloud/blog/252991/.](https://habr.com/company/1cloud/blog/252991/)
4. [https://habr.com/post/136046/.](https://habr.com/post/136046/)
5. [https://nmap.org/nsedoc/categories/vuln.html.](https://nmap.org/nsedoc/categories/vuln.html)
6. [https://linoxide.com/linux-how-to/install-security-updates-ubuntu/.](https://linoxide.com/linux-how-to/install-security-updates-ubuntu/)
7. [https://vulners.com/help.](https://vulners.com/help)
8. [https://www.ptsecurity.com/ru-ru/products/xspider/.](https://www.ptsecurity.com/ru-ru/products/xspider/)
9. <http://www.openvas.org/software.html>.
10. <https://www.kali.org/penetration-testing/openvas-vulnerability-scanning/>.
11. <http://www.irongeek.com/i.php?page=backtrack-r1-man-pages/openvasmd>.
12. <http://www.irongeek.com/i.php?page=backtrack-r1-man-pages/openvassd>.
13. <http://www.irongeek.com/i.php?page=backtrack-r1-man-pages/gsad>.
14. <https://habr.com/company/pentestit/blog/323568/>.