**Основы:**

* В чем заключается разница между активным и пассивным сбором информации?

###############################################################

Ответ:

Активный сбор информации предполагает взаимодействие с атакующей системой. Данный способ является «громким», несмотря на выбор инструмента и его настроек - в системе могут (и даже будут) остаться «следы» программы. Но такой способ получает больше информации о атакующей системе.

Пассивный сбор представляет собой OSINT и напрямую не взаимодействует с атакующей системой. Такой подход обеспечивает минимальный риск обнаружения.

################################################################

* Что такое Common Vulnerability Scoring System?

################################################################

Ответ:

Common Vulnerability Scoring System является стандартом, и он предназначен для оценки серьезности уязвимостей. Этот стандарт помогает расставлять приоритеты в ресурсах и реагировать на конкретную угрозу. Зачем это делать? Безопасность должна быть эффективной. И чтобы этого достичь, прибегают к risk management process. На одном из этапов процесса решают, как поступить с уязвимостью: принять её, попытаться исправить, смягчить угрозу и т.д. CVSS хоть и полезен, но даёт лишь базовую оценку и должен учитываться со средой безопасности организации.

##################################################################

**Общие техники и атаки:**

* Какие виды уязвимостей в DNS и ARP вы знаете? Какими способами нарушитель может их проэксплуатировать?

##################################################################

Ответ:

DNS flooding – это DDoS-атака с использованием протокола DNS. При этой атаке происходит перегрузка сервера и забивание трафика.

DNS Rogue server – при этой атаке подменяется легитимный DNS-сервер на DNS-сервер атакующего. Это может позволить перенаправлять пользователей на фишинговые сайты.

Cache poisoning – атака, которая позволяет перехватывать весь трафик жертвы. Происходит эта атака, когда DNS-сервер не может найти в своих записях запрос на определённый сайт. Когда DNS-сервер обращается к другому серверу, чтобы получить запись, то ответ получает от атакующего, который в это время непрерывно посылал ответы. После того, как DNS-сервер примет ответ атакующего, то запрос пользователя к этим записям будет вести на ресурс атакующего.

DNS-туннель используют для передачи команд, вредоносных программ и т.д. Этот туннель можно открыть через разные утилиты, например, dnscat2.

ARP spoofing позволяет атакующему перехватывать связь между сетевыми устройствами. Например, есть два устройства: хост и маршрутизатор. Атакующий может послать ответы ARP маршрутизатору, что теперь атакующий является хостом, а хосту послать, что атакующий теперь – это маршрутизатор. Это заставит и маршрутизатор, и хоста ​​подключаться к атакующему, а не друг к другу.

##################################################################

* В чем заключается разница между атакой «грубой силы» и атакой по словарю?

#############################################################

Ответ:

Dictionary attack – это быстрая атака, которая под словарём подразумевает любые цифровые ресурсы, которые содержат слова. А brute force предполагает перебор по буквам, символам, цифрам и т.д. Успех dictionary attack зависит от слов в словаре, а brute force- от вычислительной мощности компьютера.

##############################################################

* В чем заключается атака Golden Ticket (золотой билет)?

#############################################################

Ответ:

Атакующий, у которого есть хэш пароля учетной записи KRBTGT, может подделать билеты выдачи билетов Kerberos (TGT). Это позволяет запрашивать билеты службы выдачи билетов (TGS), которые обеспечивают доступ к определенным ресурсам, и получить аутентификацию для любой учетной записи в Active Directory.

#############################################################

**Атаки и уязвимости веб-приложения:**

* Какие основные виды уязвимостей веб-приложений вы знаете?

###############################################################

Ответ:

В OWASP top 10 содержатся наиболее важные риски безопасности веб-приложений:

1. Broken Access Control – возникает из-за неправильно настроенного контроля доступа. Это приводит к несанкционированному раскрытию информации, изменению или уничтожению всех данных и т.д.

2. Cryptographic Failures – возникает из-за неправильного использования (или неиспользования) криптографических алгоритмов для защиты конфиденциальной информации. Это приводит к тому, что веб-приложения случайно разглашают конфиденциальные данные.

3. Injection – возникает из-за того, что приложение интерпретирует вводимые пользователем данные как команды. Это приводит к извлечению данных из БД или OS.

4. Insecure Design – возникает из-за небезопасного проектирования и небезопасной реализацией. Это приводит к появлению дополнительных уязвимостей, например, приложение может установить большое количество попыток для ввода пароля, что позволит атакующему провести brute force и получить доступ к учётным данным.

5. Security Misconfiguration – возникает из-за неправильной настройки безопасности, например, установка политики слабых паролей. Это приводит к появлению дополнительных уязвимостей, например, XXE.

6. Vulnerable and Outdated Components – возникает из-за устаревших компонентов. Это приводит к тому, например, что веб-сервер, который не обновлялся, может содержать общеизвестную уязвимость, которую атакующий может использовать.

7. Identification and Authentication Failures – возникает из-за слабых механизмов аутентификации. Например, атакующий может пропустить форму с вводом учётных данных и сразу зайти под другим пользователем. Это приводит к несанкционированному доступу.

8. Software and Data Integrity Failures – возникает из-за использования программ или данных без каких-либо проверок целостности. Например, скачивание приложения, которое было изменено. Это приводит к различным уязвимостям, например, получение несанкционированного доступа к OS.

9. Security Logging and Monitoring Failures – возникает из-за неправильной настройки программ мониторинга и журналов или их отсутствия. Это приводит к тому, что подозрительная активность не фиксируется, а атакующий может оставаться в системе незамеченным.

10. Server-Side Request Forgery – возникает, когда атакующий может заставить веб-приложение отправлять запросы от имени сервера. Это приводит к раскрытию данных, несанкционированному доступу к внутренним системам и т.д.

################################################################

**Сценарии:**

* Представьте, что Вы нашли NTLM-хеш учетной записи, но не смогли его расшифровать. Как можно использовать эти знания для получения доступа к целевой машине?

################################################################

Ответ:

Можно создать shell с помощью программы pth-winexe от имени пользователя, например, admin и вместо пароля использовать хэш NTLM:

pth-winexe -U 'admin%NTLM\_hash' //<ip host> cmd.exe

Создать удалённое подключение, используя NT:

evil-winrm -i <ip\_host> -u Administrator -H <NT>

Если в Kerberos включен алгоритм шифрования RC4, то можно получить билет, имея только хеш пароля.

Так же хэш используется в различных техниках Pass the hash.

################################################################

* Какие меры можно предпринять для предотвращения атаки «грубой силы»?

#################################################################

Ответ:

Использовать сложный и длинный пароль;

Ограничить кол-во попыток для неправильного ввода;

Блокировать ip-адрес, у которого замечено аномально много запросов на ввод пароля;

Возвращать один и тот же код запроса (правильный или неправильный ввод), чтобы атакующему было сложнее проследить за ответом;

Использовать 2FA, где это возможно.

##################################################################

* Опишите первые попытки исследования целевой машины на возможность повышения привилегий.

###############################################################

Ответ:

Выяснить от какого пользователя открыт shell;

Узнать его права;

Узнать, какие приложения используются и их версия (версия OS была выяснена ранее)

Поиск общедоступных уязвимостей.

################################################################

**Задание №1**

На информационном ресурсе c ОС Ubuntu 18.04.4 LTS открыты следующие порты:

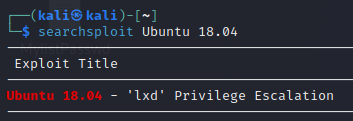
* 22/tcp - OpenSSH 7.6p1 4ubuntu0.3
* 80/tcp - Apache httpd 2.4.50

Требуется описать уязвимости, их критичность и рекомендации по устранению.

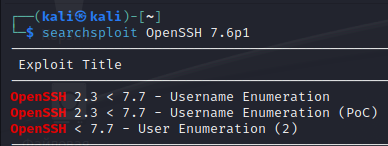
#################################################################

Ответ:

1. ОС Ubuntu 18.04.4 LTS – Privilege Escalation. Скрипт для повышения привилегий хоть и не проверен, но может попасть под категорию «CRITICAL».



2. OpenSSH 7.6 – Username Enumeration. CVSS даёт этой уязвимости базовую оценку «Medium».



3. Apache 2.4.50 – Path Traversal & RCE. Оценка CVSS колеблется от «HIGH» до «CRITICAL»



Для всех этих уязвимостей существует единое устранение – обновиться до более поздней версии.

################################################################

**Задание №2-4**

Необходимо скачать и развернуть виртуальные машины testask1-3 (https://disk.yandex.ru/d/bc2rUR-4M9iTcg). Нужно получить содержимое flag.txt (либо root.txt на одной из машин) в директории администратора.

Рассматривается эксплуатация уязвимостей по сети. Манипуляции с системой через установочный диск или загрузчик, рассматриваться не будет.

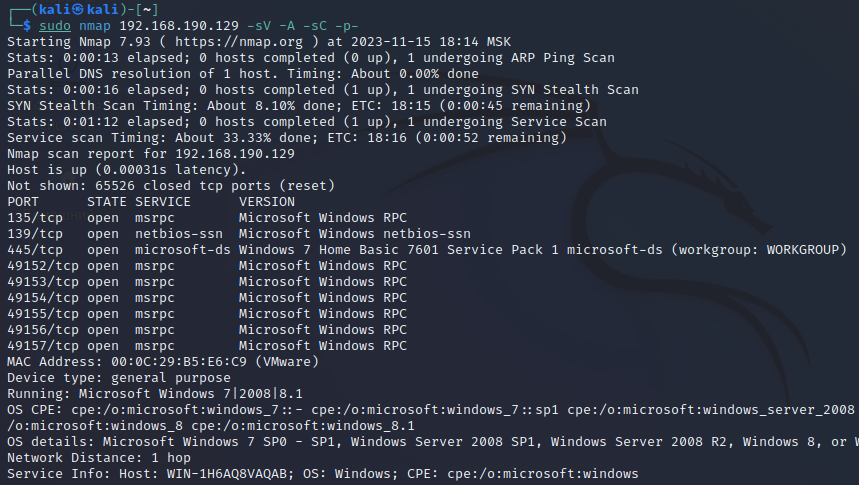
###############################################################

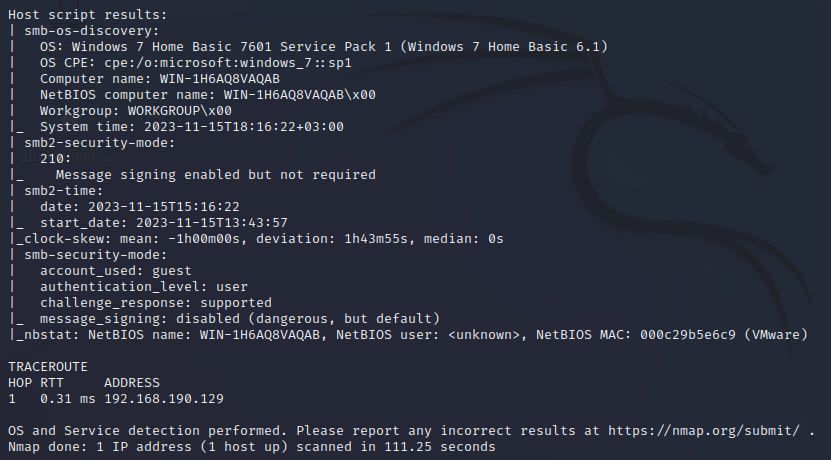
Ответ:

Task1

Забавно: я сделал все задания (это оставил напоследок), и они не вызывали никакой сложности. Но когда дошёл до практики, то даже не знал с чего начинать.

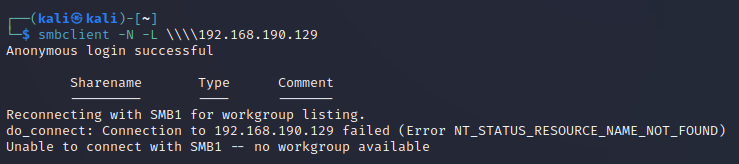
Первым делом просканировал систему (да, не аккуратно, но я знаю, что мне за это ничего не будет):





Только благодаря тому, что сработали скрипты, я понял, что в системе работает smb.

Решил посмотреть какие есть сетевые папки, но не получилось подключиться к smb1:

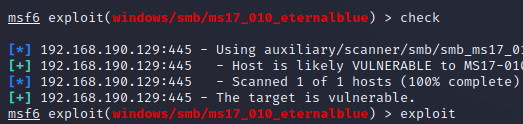


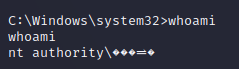
Казалось бы, вот она подсказка: windows 7 + smb1, но я забыл к какой уязвимости это относится – не каждый день с таким сталкиваюсь.

Решил вбить эту запись: “Windows 7 Home Basic 7601 Service Pack 1 microsoft-ds” в гугл, и он мне выдал: “EternalBlue”. Сразу стало понятно, что делать.

Дальше просто вставлю скриншоты:

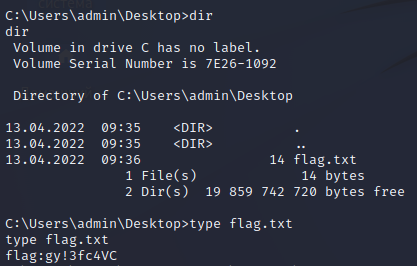






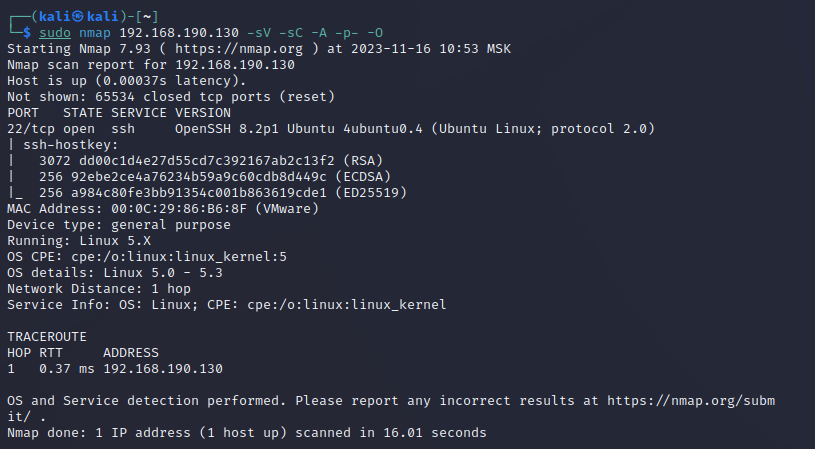




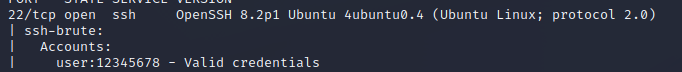


Task 2

Начинаю со сканирования системы:



Теперь знаю ssh и неполную версию ядра (5.Х). Вбивал в гугл версию ssh и получал, что версия уязвима, но как использовать эту уязвимость уже не говорилось. Решил использовать скрипты nmap:



Это был обычный пользователь:



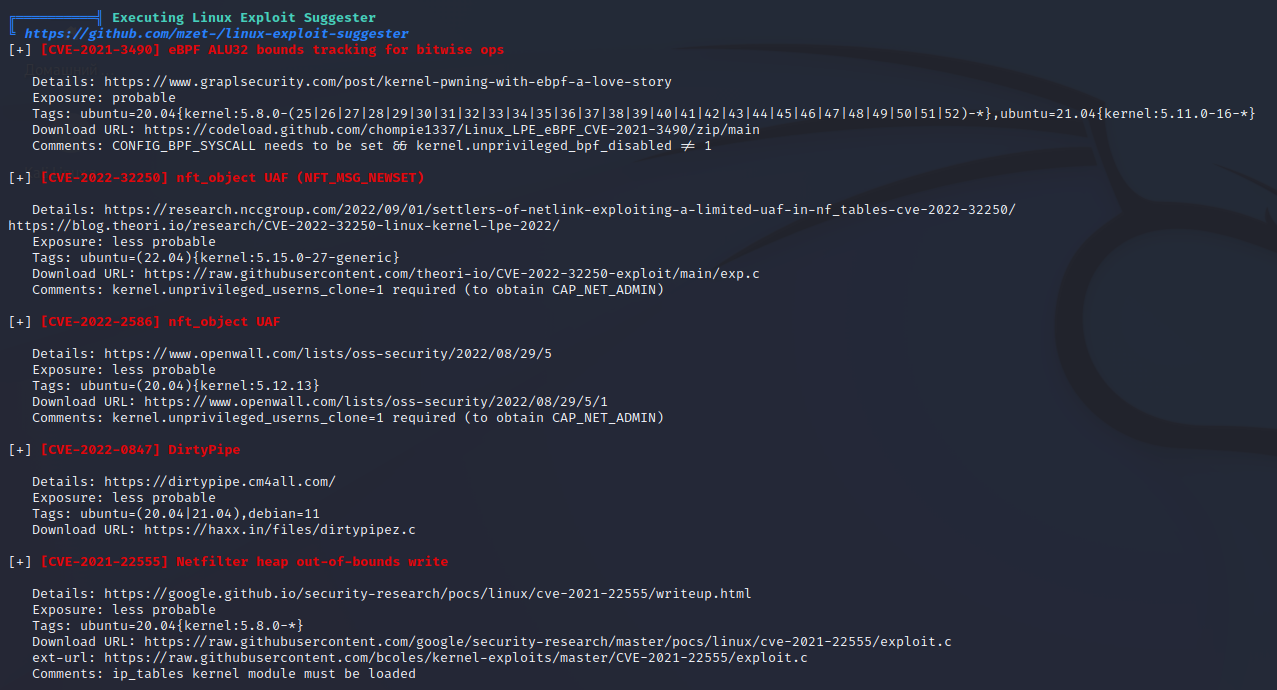
Не было возможности воспользоваться sudo:



С уязвимостями ядра раньше не сталкивался, поэтому я и не посмотрел версию ядра (зато посмотрел версию linux: Ubuntu 20.04 LTS Focal Fossa, но это ничего не дало). Не зная, что делать дальше решил использовать скрипт linPEAS, чтобы найти, куда двигаться:



Получил очень много информации, а также, какими скриптами можно воспользоваться для повышения привилегий. Но сначала хотел всё сделать без них, проверяя то, что выдал linPEAS. Это мне не помогло и пришлось смотреть, какие скрипты предлагал вывод linPEAS.

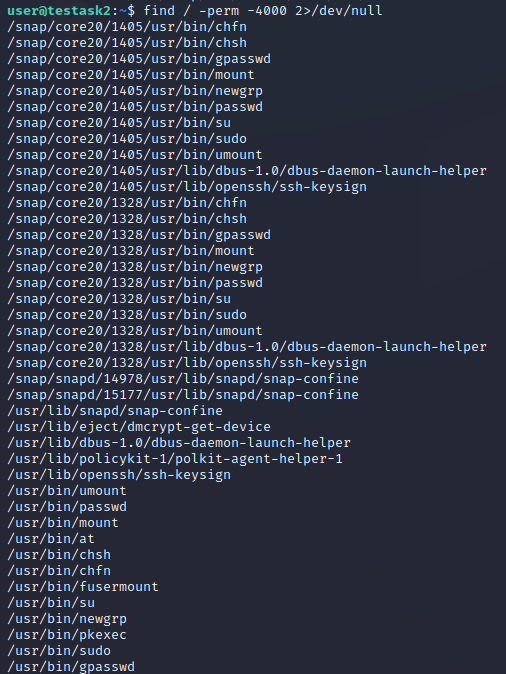


Решил сначала прочитать про DirtyPipe, т.к. слышал про DirtyCow. Узнав, что эта уязвимость была исправлена на версиях ядра 5.16.11, 5.15.25 и 5.10.102, то решил проверить свою версию:

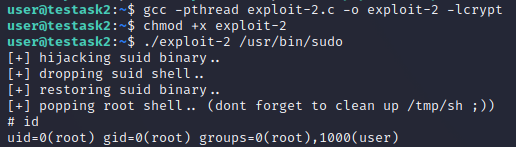


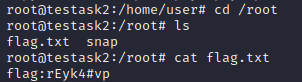
С помощью nano создал файл exploit-2.c (exploit-1 не сработал) и скопировал в него код для работы уязвимости.

Для работы этого скрипта нужно указать двоичные файлы, которые работают от root. Пришлось искать:



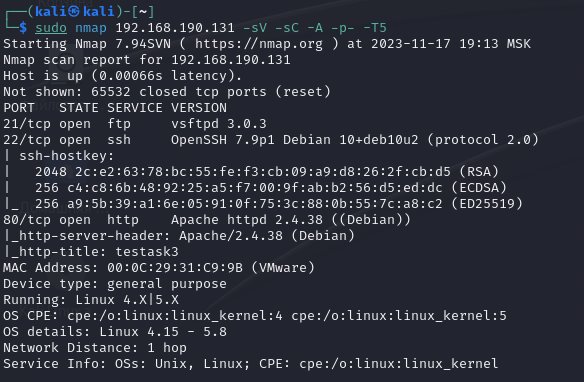
Дальше файл осталось скомпилировать и запустить:



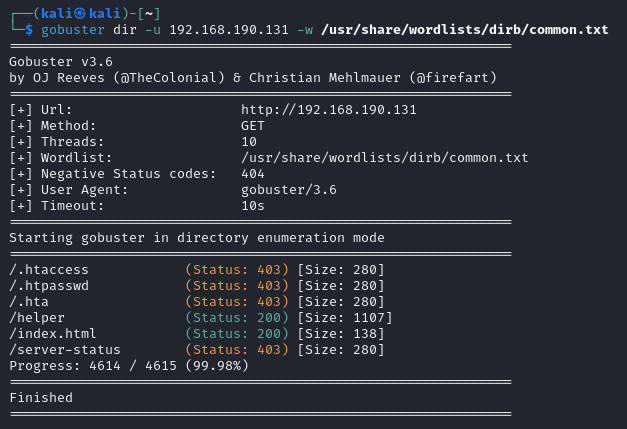


Task 3

Просканировал сеть:



На 80 порту ничего нет, но раз есть сайт, то нужно пройтись gobuster:



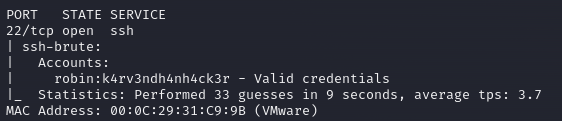
На /helper находятся УЗ на ftp:

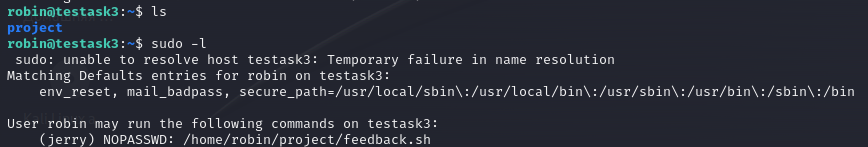


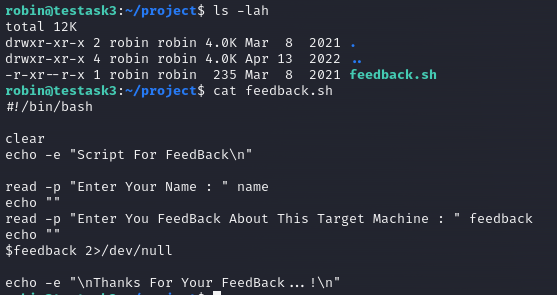
Зашёл на ftp и скачал несколько файлов:



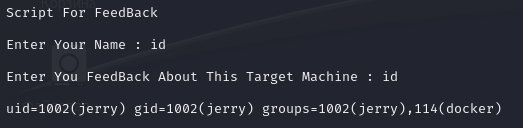
В них говорится, что пользователю robin нужно сменить пароль на тот, который находится в p\_lists.txt. Добавив эту информацию в скрипт ssh-brut, узнаю необходимый логин и пароль:



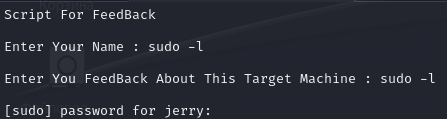




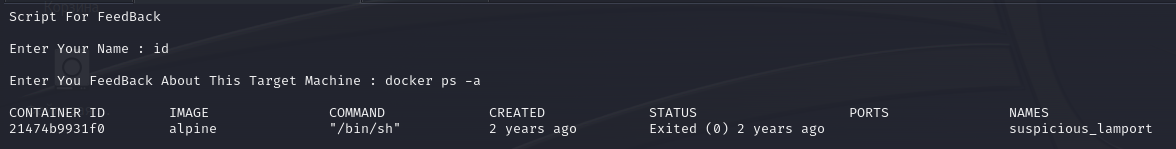
Данный скрипт не могу изменять, но его input выглядит небезопасным. Так и оказалось:



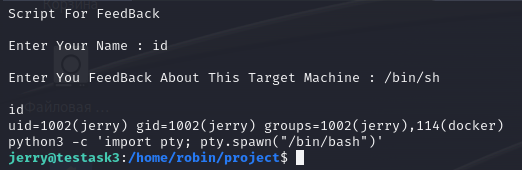
Его права не могу посмотреть:



Раз у него есть связь с докером, то решил посмотреть, какие контейнеры есть:



С помощью этого (и python) поднял shell за jerry:



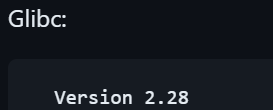
Мне кажется, это было тратой времени – никакого преимущества этот user не даёт.

По версии ядра нашёл уязвимость, которая должна помочь получить root, но у меня не получается её эксплуатировать(CVE-2019-13272). При запуске появляется ошибка: /lib/x86\_64-linux-gnu/libc.so.6: version `GLIBC\_2.34' not found. Как оказалось, нужно компилировать файл на уязвимой системе, а не там, где получится, но там нет такой программы:



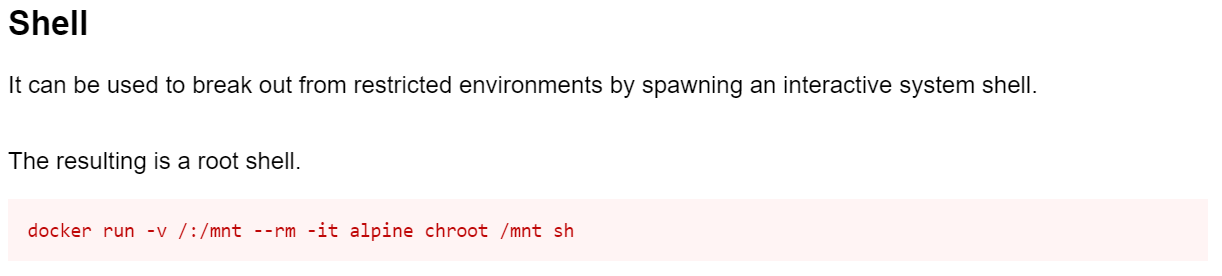


Какая должна быть версия GLIBC для успешного запуска:



Решил вернуться к пользователю jerry – может, он не бесполезен. Он добавлен в группу докер, но я думал, что использовал уже его. Начал читать, как с помощью докера можно повысить привилегии. Оказалось, могу монтировать папки в контейнер. Но так я права не получу.

Отлично. Нашёл способ повышения прав:



Забавно – тот контейнер имеет такое же имя, как у атакуемой машины.



###############################################################

**Задание №5**

Необходимо найти уязвимость в php коде:

1.

<?php

$file = $\_GET['file'];  
if (file\_exists('/home/wwwrun/'.$file.'.php')) {  
    include '/home/wwwrun/'.$file.'.php';  
}  
?>

Этот код имеет уязвимость LFI. Для предотвращения этой уязвимости нужно фильтровать пользовательский ввод.

<?php

$file = $\_GET['file'];

$fullPath = realpath('/home/wwwrun/' . $file . '.php');

if ($fullPath !== false && strpos($fullPath, '/home/wwwrun/') === 0) {

include $fullPath;

} else {

echo "Invalid file path";

}

?>

В этом коде используется функция realpath() для получения реального пути к файлу и затем проверяет, что файл находится в ожидаемом каталоге перед включением его в код.

2.

<?php

$id = $\_GET['id'];

$sql = 'SELECT name, text FROM posts WHERE id = ' . $id;

$db = new mysqli('127.0.0.1', 'root', '', 'database');

$query = $db->query($sql);

$post = $query->fetch\_assoc();

echo $post['name'];

echo '<br>';

echo $post['text'];

?>

Этот код имеет SQLi уязвимость. Здесь так же, как и в примере выше, не фильтруется пользовательский ввод. Для предотвращения уязвимости можно либо фильтровать ввод или использовать параметризированные запросы.

…

$sql = 'SELECT name, text FROM posts WHERE id = ?';

…

3.

<?php

if (PHP\_SAPI === 'cli') {

parse\_str(implode('&', array\_slice($argv, 1)), $\_GET);

}

if (NULL == $\_GET['name']) $\_GET['name'] = "Guest! ";

echo 'Hello, welcome ' . $\_GET['name'];

?>

Этот код имеет XSS уязвимость. Пользовательский ввод снова не фильтруется.

…

echo 'Hello, welcome ' . htmlspecialchars($\_GET['name']);

…

Функция htmlspecialchars() преобразует некоторые предопределенные символы в объекты HTML.